JHEP

Published for SISSA by Springer

Received: January 13, 2021
Accepted: April 5, 2021
Published: May 13, 2021

A comparison of optimisation algorithms for high-dimensional particle and astrophysics applications

The DarkMachines High Dimensional Sampling Group

Csaba Balázs,[a] **Melissa van Beekveld,**[b] **Sascha Caron,**[c,d] **Barry M. Dillon,**[e] **Ben Farmer,**[f] **Andrew Fowlie,**[g] **Eduardo C. Garrido-Merchán,**[h] **Will Handley,**[i,j] **Luc Hendriks,**[c,d] **Guðlaugur Jóhannesson,**[k,l] **Adam Leinweber,**[m] **Judita Mamužić,**[n] **Gregory D. Martinez,**[o] **Sydney Otten,**[c,p] **Roberto Ruiz de Austri,**[n] **Pat Scott,**[q,r] **Zachary Searle,**[m] **Bob Stienen,**[c,d,1] **Joaquin Vanschoren**[s] **and Martin White**[m,1]

[a] *School of Physics and Astronomy, Monash University, Melbourne, VIC 3800, Australia*
[b] *Rudolf Peierls Centre for Theoretical Physics, 20 Parks Road, Oxford OX1 3PU, U.K.*
[c] *High Energy Physics, IMAPP, Radboud University Nijmegen, Heyendaalseweg 135, 6525 AJ, Nijmegen, Netherlands*
[d] *Nikhef, Science Park 105, 1098 XG Amsterdam, Netherlands*
[e] *Institut für Theoretische Physik, Universität Heidelberg, Germany*
[f] *Bureau of Meteorology, Melbourne, VIC 3001, Australia*
[g] *Department of Physics and Institute of Theoretical Physics, Nanjing Normal University, Nanjing, Jiangsu 210023, China*
[h] *Higher Polytechnic School, Universidad Autonoma de Madrid, Francisco Tomas y Valiente 25, Madrid, Spain*
[i] *Cavendish Laboratory, University of Cambridge, JJ Thomson Avenue, Cambridge, CB3 0HE, U.K.*
[j] *Kavli Institute for Cosmology, Madingley Road, Cambridge, CB3 0HA, U.K.*
[k] *Science Institute, University of Iceland, Dunhaga 7, IS-107 Reykjavik, Iceland*
[l] *Nordita, KTH Royal Institute of Technology and Stockholm University, Roslagstullsbacken 23, SE-106 91 Stockholm, Sweden*
[m] *ARC Centre for Dark Matter Particle Physics, Department of Physics, University of Adelaide, Adelaide, SA 5005, Australia*
[n] *Instituto de Física Corpuscular, IFIC-UV/CSIC, Carrer del Catedrátic José Beltrán Martinez, 2, Valencia, Spain*
[o] *Physics and Astronomy Department, University of California, Los Angeles, CA 90095, U.S.A.*

[1] Corresponding author.

https://doi.org/10.1007/JHEP05(2021)108

JHEP05(2021)108

[p] *Gravitation Astroparticle Physics Amsterdam (GRAPPA),*
Institute for Theoretical Physics Amsterdam and Delta Institute for Theoretical Physics,
University of Amsterdam,
Science Park 904, 1098 XH Amsterdam, The Netherlands
[q] *School of Mathematics and Physics, The University of Queensland,*
St. Lucia, Brisbane, QLD 4072, Australia
[r] *Department of Physics, Imperial College London, Blackett Laboratory,*
Prince Consort Road, London SW7 2AZ, U.K.
[s] *Eindhoven University of Technology, Groene Loper 5, 5612 AZ Eindhoven, Netherlands*

E-mail: martin.white@adelaide.edu.au, bstienen@science.ru.nl

Abstract: Optimisation problems are ubiquitous in particle and astrophysics, and involve locating the optimum of a complicated function of many parameters that may be computationally expensive to evaluate. We describe a number of global optimisation algorithms that are not yet widely used in particle astrophysics, benchmark them against random sampling and existing techniques, and perform a detailed comparison of their performance on a range of test functions. These include four analytic test functions of varying dimensionality, and a realistic example derived from a recent global fit of weak-scale supersymmetry. Although the best algorithm to use depends on the function being investigated, we are able to present general conclusions about the relative merits of random sampling, Differential Evolution, Particle Swarm Optimisation, the Covariance Matrix Adaptation Evolution Strategy, Bayesian Optimisation, Grey Wolf Optimisation, and the PyGMO Artificial Bee Colony, Gaussian Particle Filter and Adaptive Memory Programming for Global Optimisation algorithms.

Keywords: Phenomenology of Field Theories in Higher Dimensions, Supersymmetry Phenomenology

ArXiv ePrint: 2101.04525

JHEP05(2021)108

Contents

1 Introduction

Typical theories in various branches of science, such as particle physics, particle astrophysics and astrophysics, are formulated as parametric models. To make predictions in these models, one needs to specify the values of a set of free numerical parameters. Comparison with experiment, in turn, can constrain the values of these parameters and single out a theory that matches observation the best. Such parameter estimation and model selection are fundamental components of the scientific method in the physical sciences that we hope will lead us to the best description of nature.

JHEP05(2021)108

In the modern era, the exploding size of experimental data sets, coupled with complicated theories, has introduced a substantial computational complexity in parameter extraction and model selection. It is frequently necessary to perform computationally expensive simulations of experiments, and further problems result from the moderately large dimensionality of typical beyond-Standard Model physics models, whose parameter multiplicity often ranges up to $\mathcal{O}(100)$. In parameter spaces of this size, sampling randomly or on a uniform grid is both inefficient and unlikely to lead to robust statistical conclusions [1].

We have entered a period in fundamental physics where one experiment is unlikely to unambiguously determine the next theory of particle physics or cosmology. Consequently, we must combine clues from many different branches of observation. Fortunately, at the same time, exponentially increasing computing power has allowed scientists to become more ambitious both in the scope of observation and theoretical calculation. Thus, parameter extraction and model comparison can be performed at an extraordinary scale to find the theory that best describes the physical world, if one employs efficient and ingenious sampling algorithms.

The likelihood function is a key quantity in statistical inference (see e.g., ref. [2] for a pedagogical introduction), as it tells us the probability of the observed experimental data for a particular set of model parameters. If we are considering data from several experiments, the likelihood function may often be written as a product of likelihoods, one for each of the individual experiments, if the individual experiments are independent. In frequentist statistics, one can obtain consistent estimators for the values of the parameters of the model by finding the set of parameters that maximises the likelihood, and use the maximum likelihood itself to construct a test-statistic to perform statistical tests (see e.g., ref. [3] for an introduction to likelihood-based tests in particle physics). The difficulty is that the likelihood function is rarely known as a simple function of the original parameters and so we cannot find the maximum analytically. In fact, although in our setting we assume a tractable likelihood, evaluating the likelihood may still involve non-differentiable forward simulations of experiments (see e.g., ref. [4]) and so even derivatives are unavailable. We are thus forced to use derivative-free numerical optimisation algorithms (see ref. [5] for a review) to explore the likelihood function.

Furthermore, likelihood functions of interest often contain multiple modes, i.e., several distinct local maxima. In this setting, exploring the likelihood function and locating the global maximum may be extremely challenging, as we risk getting stuck in a local maximum. For this reason, we focus on stochastic algorithms that, e.g., step out of a local maximum with a particular probability, and neglect local optimisers commonly used in physics [6]. The simplest such approach is repeated random sampling from the entire parameter space (followed by picking either the highest or lowest value found.). This, however, is known to be deficient for two reasons. First, as the dimension of the parameter space increases, the number of samples that need to be drawn increases exponentially if the same point density is to be maintained. In practice this would thus come with an exponentially higher demand for computational power, which is worsened by the fact that the function evaluations are typically already costly themselves. Second, it is highly inefficient in most physical examples, as the high likelihood regions of the parameter space

JHEP05(2021)108

usually occupy a very small region of the total multidimensional volume. The past decades have thus seen the development of a series of novel sampling and optimisation procedures, particularly metaheuristic ones, many of which have been utilised in particle astrophysics applications [7–102].

The purpose of this paper is to survey a wide range of optimisation techniques that, to the best of our knowledge, have not received mainstream use in particle astrophysics applications. The different techniques are explored by different authors of this paper in the form of the following challenge: use your optimisation technique of choice to find the optima of a set of reference functions, including both analytic examples and a 12-dimensional parameter space representing a supersymmetric model called the phenomenological MSSM7 [103] (a popular theory of beyond-Standard Model Particle Physics). Apart from the common set of reference functions and the use of a common test framework, there was no common tuning of the free parameters of the techniques. Although this introduces a human factor in the experiments, we believe this is representative of a real-life application of any one of the explored methods. The work was completed within the DarkMachines community,[1] which aims to develop new approaches for dark matter research thorough closer collaboration with machine learning and data science experts. It is worth noting that the algorithms we compare of course have many uses beyond maximising likelihood functions, such as minimisation of fine-tuning or the optimisation of the hyperparameters of an algorithm. Our ultimate aim is to provide a self-contained overview of optimisation methods that can be used as a reference by researchers working in the physical sciences. For the purposes of this publication, we developed a testing framework in Python with interfaces to codes representing each of the optimisation techniques we use below.

This paper is structured as follows. In section 2, we define our optimisation problem in more detail, and provide a description of each of the techniques used. We describe the test functions that we use for our comparative studies in section 3, and detail the results in section 4. Finally, we present conclusions in section 5, and describe our Python framework for implementing the scanning techniques in appendix A. The best found solutions for each investigated function and algorithm, together with the corresponding algorithm's hyperparameters, can be found in appendix B.

2 Optimisation

The problem that we address in this review is the following. Given a deterministic function $f : \mathbb{R}^n \to \mathbb{R}$, defined over a domain of interest given by lower and upper bounds on the function parameters, what is the optimum of the function? In particle astrophysics applications, the function can represent the likelihood of observed data given a particular physical theory, and the optimum then gives the maximum likelihood estimate of the parameters, which is an important quantity in frequentist inference. It is in fact more usual to minimise the negative log-likelihood function, rather than maximise the likelihood.

Optimisation techniques can be divided into categories, based on whether they require a knowledge of the derivatives of the function or not (either analytical or numerical). Since

[1] http://www.darkmachines.org.

JHEP05(2021)108

derivative information is not always available in the physical sciences, we focus on techniques that only require evaluations of the likelihood function itself. Arguably the most challenging optimisation problems in particle astrophysics applications arise in global fits of beyond-Standard Model physics models. Popular techniques for performing frequentist inference on dark matter models have included *Markov Chain Monte Carlo* techniques (see e.g., ref. [104]) and *nested sampling* [105] which, although designed for Bayesian computation, can be repurposed for frequentist studies [7–22, 24, 26–28, 30–95]. In recent years, *genetic algorithms* and, to an even greater extent, Differential Evolution, have proven capable of adequately exploring very complex likelihood functions in multiple theories of dark matter [23, 103, 106–110]. In selecting techniques for this study, we have focused on global optimisers that are expected to provide comparable performance to Differential Evolution. We include the Differential Evolution implementation previously studied in [111] in order to benchmark the performance of our newly-explored techniques. The full list of algorithms that we explore is as follows.

2.1 Optimisation algorithms

2.1.1 Differential Evolution

Differential Evolution [DE; 112–115] is a population-based heuristic optimisation strategy belonging to the class of *evolutionary algorithms*. DE does not rely on derivatives of the function being optimised, and is often the algorithm of choice for highly multimodal or otherwise poorly-behaved objective functions.

DE consists of evolving a population of NP individuals or 'target vectors' $\{\mathbf{X}_i^g\}$, of specific points in the parameter space, for a number of generations. Here i refers to the ith individual, and g corresponds to the generation of the population. The initial generation is generally selected randomly within the parameter intervals to be sampled.

One generation is evolved to the next via three main steps: mutation, crossover and selection. The simplest variant of the algorithm is known as rand/1/bin; the first two parts of the name refer to the mutation strategy (random population member, single difference vector), and the third to the crossover strategy (binomial).

Mutation proceeds by identifying an individual $\mathbf{X}_i$ to be evolved, and constructing one or more donor vectors $\mathbf{V}_i$ with which the individual will later be crossed over. In the rand/1 mutation step, three unique random members of the current generation $\mathbf{X}_{r1}$, $\mathbf{X}_{r2}$ and $\mathbf{X}_{r3}$ are chosen (with none equal to the target vector), and a single donor vector $\mathbf{V}_i$ is constructed as

$$\mathbf{V}_i = \mathbf{X}_{r1} + F(\mathbf{X}_{r2} - \mathbf{X}_{r3}), \tag{2.1}$$

with the scale factor F being a parameter of the algorithm. A more general mutation strategy known as rand-to-best/1 also allows some admixture of the current best-fit individual $\mathbf{X}_{\rm best}$ in a single donor vector,

$$\mathbf{V}_i = \lambda\mathbf{X}_{\rm best} + (1-\lambda)\mathbf{X}_{r1} + F(\mathbf{X}_{r2} - \mathbf{X}_{r3}), \tag{2.2}$$

according to the value of another free parameter of the algorithm, λ.

JHEP05(2021)108

Crossover then proceeds by constructing a trial vector $\mathbf{U}_i$, by selecting each component (parameter value) from either the target vector, or from one of the donor vectors. In simple binomial crossover (the bin of rand/1/bin), this is controlled by an additional algorithm parameter Cr. For each component of the trial vector $\mathbf{U}_i$, a random number is chosen uniformly between 0 and 1; if the number is greater than Cr, the corresponding component of the trial vector is taken from the target vector; otherwise, it is taken from the donor vector. At the end of this process, a single component of $\mathbf{U}_i$ is chosen at random, and replaced by the corresponding component of $\mathbf{V}_i$ (to make sure that $\mathbf{U}_i \neq \mathbf{X}_i$).

Selection simply faces the target vector $\mathbf{X}_i$ off against the trial vector $\mathbf{U}_i$, with the vector returning the best value of the objective function retained for the next generation. In this way, each member of a generation is pitted against exactly one trial vector in each generation step.

A widely-used variant of simple rand/1/bin Differential Evolution is so-called jDE [116], where the parameters F and Cr are optimised on-the-fly by the Differential Evolution algorithm itself, as if they were regular parameters of the objective function. An even more aggressive variant known as λjDE [111] is the self-adaptive equivalent of rand-to-best/1/bin, where F, Cr and λ are all dynamically optimised.

In this paper, we run two different software implementations of Differential Evolution. The first is the open-source implementation of the λjDE algorithm contained in the Diver package.[2] We use this via the pyScannerBit interface to the ScannerBit package of the GAMBIT code for beyond-Standard Model global statistical fits [106, 111, 117]. We also run the jDE [116] and iDE [118] algorithms implemented in the PyGMO package [119]. In doing so, we have varied the number of generations and the parameter adaptation scheme which is used to optimise the weight coefficient and the crossover probability.

2.1.2 Particle Swarm Optimisation

Particle Swarm Optimisation [PSO; 120, 121] is another population-based evolutionary algorithm that does not make use of derivatives. Here, each member of the population of parameter samples ('the swarm') is also given a velocity. In each generation step, the positions of other particles in the swarm are used to update each particle's velocity. The position of each particle is updated by allowing it to move along its velocity vector for a fixed amount of time.

The standard velocity update for particle i in generation g is

$$\mathbf{v}_i^{g+1} = \omega \mathbf{v}_i^g + \phi_1 r_1 (\mathbf{x}_{i,\mathrm{pb}} - \mathbf{x}_i^g) + \phi_2 r_2 (\mathbf{x}_{\mathrm{gb}} - \mathbf{x}_i^g), \tag{2.3}$$

such that $\mathbf{x}_i^{g+1} = \mathbf{x}_i^g + \mathbf{v}_i^g$. Here r_1 and r_2 are uniform random numbers between 0 and 1, $\mathbf{x}_{i,\mathrm{pb}}$ is the ith particle's personal best-fit position so far (i.e. in any generation), $\mathbf{x}_{\mathrm{gb}}$ is the global best-fit position so far (i.e. by any particle, in any generation), and ω, ϕ_1 and ϕ_2 are free parameters of the algorithm.

In this paper, we will make use of a self-adaptive variant inspired by jDE [116] that we will refer to as j-Swarm, where ω and/or ϕ_1 and ϕ_2 can be dynamically optimised in the

[2] https://diver.hepforge.org.

JHEP05(2021)108

course of a run by treating them as parameters of the objective. The implementation that we use is bundled in ScannerBit, and will be released within GAMBIT and described in a forthcoming GAMBIT publication.

2.1.3 CMA-ES

The Covariance Matrix Adaptation (CMA) Evolution Strategy (ES) is another evolutionary optimisation algorithm based on the idea of natural selection [122]. From an initial point in the n-dimensional parameter space, $\mathbf{x}^{(0)}$, a set of λ new points (called a population) are sampled from a multivariate normal distribution

$$\mathbf{x}_k^{(g+1)} \sim \mathcal{N}\left(\mathbf{x}^{(g)}, \left(\sigma^{(g)}\right)^2 \mathbf{C}^{(g)}\right) \tag{2.4}$$

with covariance matrix $\left(\sigma^{(g)}\right)^2 \mathbf{C}^{(g)}$, where $k = 1, \ldots, \lambda$ and g counts the number of generations. The optimisation function is then evaluated at all $\mathbf{x}_k^{(g+1)}$. The obtained values are used to sort the points and the best μ points (the parents) used to calculate

$$\mathbf{x}^{(g+1)} = \sum_{j=1}^{\mu} w_j x_j^{(g)} \tag{2.5}$$

where $w_j > 0$ are weights with $\sum_{j=1}^{\mu} w_j = 1$ and j is the sorted index running from best to worst point.

For optimum performance of the algorithm, the step size $\sigma^{(g)}$ and the matrix $\mathbf{C}^{(g)}$ should be updated to maximize the probability that the new generation is closer to the minimum of the objective function. The optimal update has been found to be given with

$$\begin{aligned}
\mathbf{p}_c^{(g+1)} &= (1 - c_c)\mathbf{p}_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{\text{eff}}}\, \frac{\mathbf{x}^{(g+1)} - \mathbf{x}^{(g)}}{\sigma^{(g)}}, \\
\mathbf{C}^{(g+1)} &= (1 - c_{\text{cov}})\mathbf{C}^{(g)} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}} \mathbf{p}_c^{(g+1)} \mathbf{p}_c^{(g+1)^T} \\
&\quad + c_{\text{cov}} \left(1 - \frac{1}{\mu_{\text{cov}}}\right) \sum_{j=1}^{\mu} w_j \left(\frac{\mathbf{x}_j^{(g+1)} - \mathbf{x}^{(g)}}{\sigma^{(g)}}\right) \left(\frac{\mathbf{x}_j^{(g+1)} - \mathbf{x}^{(g)}}{\sigma^{(g)}}\right)^T .
\end{aligned} \tag{2.6}$$

Here, $\mathbf{p}_c^{(g)}$ is a cumulative path, storing information about the direction taken during previous steps, $c_c < 1$ is the learning rate for the cumulative path, $c_{\text{cov}} < 1$ is the learning rate for the covariance matrix and $\mu_{\text{cov}} \geq 1$ controls the ratio between cumulation and rank-μ updates. The parameter $\mu_{\text{eff}}^{-1} = \sum_{j=1}^{\mu} w_j^2$ represents the effective selection mass. The cumulation update adapts the matrix to the large scale gradient of the optimisation function, while the rank-μ update adapts to the local gradient.

The optimal update for the step size is based on the absolute length of the cumulative path. If consequent steps are taken in the same direction, the length is expected to be large, while for steps in random direction, the length is shorter. In the former case, fewer and longer steps can be taken, while in the latter the step size should be decreased. The

JHEP05(2021)108

update is therefore

$$\begin{aligned}\mathbf{p}_\sigma^{(g+1)} &= (1-c_\sigma)\mathbf{p}^{(g)} + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\,\mathbf{C}^{(g)^{-1/2}}\left(\frac{\mathbf{x}^{(g+1)}-\mathbf{x}^{(g)}}{\sigma^{(g)}}\right)\\ \sigma^{(g+1)} &= \exp\left(\frac{c_\sigma}{d_\sigma}\left[\frac{|\mathbf{p}_\sigma^{(g+1)}|}{|\mathcal{N}(0,\mathbf{I})|}\right]\right)\end{aligned} \tag{2.7}$$

with $\mathbf{C}^{(g)^{-1/2}} = \mathbf{B}^{(g)}\mathbf{D}^{(g)^{-1}}\mathbf{B}^{(g)^T}$ where $\mathbf{C}^{(g)} = \mathbf{B}^{(g)}\mathbf{D}^{(g)^2}\mathbf{B}^{(g)^T}$ is the eigenvalue decomposition of $\mathbf{C}^{(g)}$. The adaptation speed is controlled by the learning rate c_σ and the damping parameter d_σ.

The CMA-ES is an invariant and stationary optimisation algorithm, meaning that its tuning parameters are insensitive to the objective function and depend almost exclusively on the dimensionality of the parameter space. Only a brief overview of the algorithm has been given here, for details and explanations see ref. [122]. The implementation used in this work is from the pycma package.[3]

2.1.4 Bayesian Optimisation

Bayesian Optimisation [BO; 123–125] is a set of techniques that attempts to find the optimum $\mathbf{x}^*$ of an objective function $f(\mathbf{x})$ with the minimum number of function evaluations, which is particularly useful when the function is computationally expensive to evaluate. It works by explicitly approximating the objective function $f(\mathbf{x})$ with a probabilistic regression model, called a surrogate model, that can predict the outcome of yet unseen samples to make a more informed decision of which samples to evaluate next. The initial surrogate model is trained on a set of random samples of the objective function, or a set of samples selected by any other sampling technique. The surrogate model needs to be probabilistic, and popular choices are Gaussian processes or probabilistic ensembles. Every further sample of the objective function $f(\mathbf{x})$ counts as a training point $\mathbf{x}$, continuously updating the surrogate model to a new posterior distribution that, after a given number of samples $\mathcal{D}\{(\mathbf{x}_i, y_i)\}$, gives our best belief of what the objective function $f(\mathbf{x})$ looks like. It can also provide some uncertainty at each point in the parameter space by using Gaussian likelihoods. An acquisition function $\alpha(\mathbf{x})$ is used to choose where to sample next, taking the latest posterior of the surrogate model as an input. The acquisition function $\alpha(\mathbf{x})$ is easy to evaluate and can be sampled with techniques such as Thompson sampling. In this way, cheap samples of the surrogate model are used to guide the sampling, rather than expensive samples of the objective function $f(\mathbf{x})$ itself. To avoid getting stuck in local minima, the acquisition function trades off exploration and exploitation, exploring regions of high expected outcome and those where the model shows high uncertainty, and hence requires more training data than is currently available. The result of this procedure is that fewer samples are needed to find the optimum $\mathbf{x}^*$ of the objective function $f(\mathbf{x})$, but more computation is required for predicting each next sample to try. More formally, the method consists of the following steps.

[3] https://github.com/CMA-ES/pycma.

JHEP05(2021)108

Step 1: define a surrogate model. Assuming that we use a Gaussian process (GP), we also need to choose a prior distribution and a covariance function, or kernel, that defines the shape of the regression curves. The surrogate model can be written as $f(\mathbf{x}) \sim GP(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. We can assume a normal prior with $\mu(\mathbf{x}) = 0$ without loss of generality. A popular choice for the kernel k is the radial basis function, also known as the square exponential kernel, in which the length scale λ and signal variance σ^2 control how flexible or flat the surrogate model can be:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \cdot \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\lambda^2}\right), \quad \lambda > 0. \tag{2.8}$$

Step 2: choose an acquisition function. A number of acquisition functions are commonly used, each defining a specific way to trade off exploration and exploitation. Given a surrogate model trained on n samples that can return, for every input $\mathbf{x}$, the predicted mean $\mu(\mathbf{x})$ and standard deviation $\sigma(\mathbf{x})$, as well as the current best value f^* and a tuneable parameter ψ which balances exploration and exploitation, we can describe the following popular choices:

1. Maximum probability of improvement (MPI):

$$a_{\text{MPI}}(\mathbf{x}) = \Phi(\gamma(\mathbf{x})), \qquad \text{where } \gamma(\mathbf{x}) = \frac{\mu(\mathbf{x}) - f^* - \psi}{\sigma(\mathbf{x})} \tag{2.9}$$

2. Expected improvement (EI):

$$a_{\text{EI}}(\mathbf{x}) = (\mu(\mathbf{x}) - f^*)\Phi(\gamma(\mathbf{x})) + \sigma(\mathbf{x})\phi(\gamma(\mathbf{x})) \tag{2.10}$$

3. Upper confidence bound (UCB):

$$a_{\text{UCB}}(\mathbf{x}) = \mu(\mathbf{x}) - \psi\sigma(\mathbf{x}) \tag{2.11}$$

In these equations Φ and ϕ are the CDF and PDF of the standard normal distribution, respectively.

The acquisition function $\alpha(\mathbf{x})$ is sampled using, for instance, Thompson sampling, and the sample with the highest acquisition score will be evaluated next on the actual optimisation function. After evaluation, this sample becomes a new training point, the surrogate model is updated, and the next sample is selected. This process is iterated until an acceptable result is reached, a certain budget is exhausted, or when all acquisition scores fall below a predefined threshold. The final recommendation of the optimisation process is the best observed result $\mathbf{x}^*$ or the optimisation of the mean of the updated posterior distribution on all observations $\mathcal{D}\{(\mathbf{x}_i, y_i)\}$.

2.1.5 Trust Region Bayesian Optimisation

Standard Bayesian Optimisation suffers from scalability issues in high-dimensional problems. This is mainly due to the implicit homogeneity assumption of most surrogate models, and

JHEP05(2021)108

an overemphasis on exploration in the used acquisition functions. The acquisition function $\alpha(\mathbf{x})$, also becomes difficult to optimise in high-dimensional problems as it has the same number of dimensions as the number of dimensions of the input space.

For instance, the commonly used Gaussian process surrogates in Bayesian Optimisation assume a constant length scale λ and signal variance σ in the search space. This is often not the reality of high dimensional functions, which tend to be flat in most of the space between local or global optima. Moreover, in high-dimensional spaces, samples are few and far between, meaning that the surrogate will exhibit high uncertainty and cause the acquisition function $\alpha(\mathbf{x})$ to focus predominantly on exploration instead of exploitation. This will harm the performance of Bayesian Optimisation.

To overcome those issues, the Trust Region Bayesian Optimisation (TuRBO) algorithm [126] fits a set of local models and determines how to allocate samples from those such that the global optimum $\mathbf{x}^*$ is found most efficiently. It performs a collection of simultaneous local optimisation executions using independent Gaussian processes.

Through this procedure, each Gaussian process enjoys the typical benefits of Bayesian modeling and these local surrogates allow for heterogeneous modeling of the objective function $f(\mathbf{x})$ without suffering from over-exploration. In order to optimise all surrogates, TuRBO leverages an implicit multi-armed bandit strategy at each iteration to allocate samples between these local areas and thus decide which local optimisation runs to continue.

Each Gaussian process resides in a Trust Region, a hyperrectangle centered at the best solution found so far, $\mathbf{x}^*$, under a surrogate model with a base side length L that is a function of the length scales λ of the Gaussian process: $L_i = \lambda_i L/(\prod_{j=1}^{d} \lambda_j)^{1/d}$. The local optimisation runs use a batch acquisition function that is restricted to select points $\mathbf{X}$ that lie within the Trust Region hyperrectangle.

The base side length L evolves during the optimisation process. If L contains all the input space $\mathcal{X}$ the algorithm becomes standard Bayesian Optimisation. The trade-off between exploration, large L, and exploitation of good solutions, small L, becomes critical. A popular criterion is to double L in size after τ results better than $\mathbf{x}^*$, and halve the size in the other case.

TuRBO maintains m trust regions simultaneously, selecting `max_eval` candidates drawn from the union of all trust regions and updating all local optimisation problems for which candidates were drawn. TuRBO gets the i^{th} candidate from all the trust regions by drawing a sample of the posterior mean function from all the Gaussian Processes per trust region and getting the minimum from them all, which is a greedy Thompson Sampling approach:

$$x_i = \min_{l} \min_{x \in \text{trust region}} f_l \,, \tag{2.12}$$

where f_l is a sample from the GP corresponding to trust region l.

2.1.6 Grey Wolf Optimisation

The Grey Wolf Optimisation algorithm [127] falls under the category of swarm intelligence algorithms, and takes inspiration from the hunting mechanism and leadership hierarchies in packs of grey wolves. Like other swarm intelligence algorithms there are two main phases:

JHEP05(2021)108

exploration and exploitation. The algorithm takes inspiration from the behaviour of grey wolves here through their tracking and attacking of prey, in which the social hierarchy of the pack of grey wolves also plays an important role. In the grey wolf analogy the wolves are associated with the candidate solutions in the swarm and the prey is associated with the best solution to the optimisation problem. The search agents in the algorithm are assigned to one of four categories, α, β, δ, or ω, which are meant to imitate the social hierarchy of the grey wolves. The fittest solution is assigned to α, the second fittest to β, the third fittest to δ, and the rest to ω. During the optimisation the search for the optimal solution is usually led by the α solution, however the β and δ also have influence.

The algorithm is initiated by assigning the search agents to random solutions in the search space, and these candidate solutions are assigned to the categories based on their fitness. The positions of each search agent are then updated, with the updates containing stochastic components as well as influence from the positions of the α, β, and δ search agents. In order to model the search strategy of the search agents the following definitions are required:

$$\begin{aligned} \vec{A} &= 2\vec{a} \cdot \vec{r}_1 - \vec{a} \\ \vec{C} &= 2\vec{r}_2. \end{aligned} \tag{2.13}$$

These vectors have a length equal to the dimension of the search space. The vectors $\vec{A}$ and $\vec{C}$ will be used to update the positions of the search agents, where $\vec{r}_{1,2}$ and vectors of random numbers in $[0, 1]$, and $\vec{a}$ has components which decrease linearly from 2 to 0 over the course of the iterations. Vectors capturing the distance between each agent and the three fittest agents are defined as:

$$\begin{aligned} D_\alpha &= |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}| \\ D_\beta &= |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}| \\ D_\delta &= |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}|. \end{aligned} \tag{2.14}$$

The positions of each agent are then updated according to:

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{2.15}$$

where t labels the current iteration, and:

$$\begin{aligned} \vec{X}_1 &= \vec{X}_\alpha - \vec{A}_1 \cdot D_\alpha \\ \vec{X}_2 &= \vec{X}_\beta - \vec{A}_2 \cdot D_\beta \\ \vec{X}_3 &= \vec{X}_\delta - \vec{A}_3 \cdot D_\delta. \end{aligned} \tag{2.16}$$

The form of these updates are such that the α, β, and δ agents estimate the position of the optima by encircling it, and the ω agents search randomly around this position. The sequence of updates is continued until some condition on the fitness of the optimal solution is met, or for a fixed number of iterations. The only hyper-parameters in this algorithm are the number of agents to use, which should be at least 4. It is possible to introduce other hyper-parameters by varying the possible range of the numbers in the vector $\vec{a}$ also, but this is not part of the minimal set-up described in [127].

JHEP05(2021)108

2.1.7 PyGMO Artificial Bee Colony

The Artificial Bee Colony algorithm is inspired by the intelligent behaviour of honey bees in their search for the right food sources [128, 129]. The algorithm has an automated mechanism to balance exploration and exploitation. In the experiments performed for this paper we use the implementation of the Artificial Bee Colony in PyGMO [119], for which the pseudocode can be found in Listing 2 of ref. [130]. Here we restrict ourselves to a more conceptual explanation of the algorithm.

The Artificial Bee Colony algorithm keeps track of SN active data points x_s, where s runs from 1 to SN. The locations of these data points are uniformly initialised and the function value f_s for each of these data points is evaluated. In the bee-analogy these data points can be seen as the locations of food sources and the function value can be interpreted as being related to the food gain from each of these sources. The goal of the algorithm is then to find the food source (i.e. data point) with the highest gain (i.e. best function value).

Finding this data point is done iteratively, where each iteration consists of three phases. In the first phase each active data point x_i is used as reference to explore a new location v_i. The proposal location v_i is calculated by moving, for each dimension j, in the direction of another, randomly selected active data point x_k:

$$v_{i,j} = x_{i,j} + \varphi_{ij}(x_{i,j} - x_{k,j}), \tag{2.17}$$

where φ_{ij} is a uniform random number between 0 and 1. For the proposal v_i the functional value f'_s is calculated and if this value is better than f_s, x_i and f_s will be replaced by v_i and f'_s respectively. The old location and function value are kept if this test fails.

For each data point the number of failed update attempts is kept track of. This information can be used to give up on locations x_i that fail to be updated for many subsequent iterations. This is done in the second phase, where these dead data points (as determined with respect to some user-configured threshold) are reinitialised uniformly in the parameter space.

These first two steps can be seen as a form of exploration: the parameter space is explored by sampling new data points and evaluating their function values. The third — and last — step of each iteration is a form of exploitation, in which equation (2.17) is used to perform updates on the active data points. Which active data points are updated is determined by assigning to each data point x_i a fitness-score.[4] This score is dependent on the functional value f_i of each data point:

$$\text{fitness}_i = \begin{cases} (1 + f_i)^{-1} & f_i \geq 0 \\ 1 + |f_i| & f_i < 0 \end{cases}. \tag{2.18}$$

This score is then used to calculate an update probability for each data point

$$p_i = \frac{\text{fitness}_i}{\sum_i \text{fitness}_i}. \tag{2.19}$$

[4]The fitness-score presented here is the one for minimisation problems.

JHEP05(2021)108

The update attempts are distributed over the active data points using Bernoulli experiments with respect to these probabilities. Just like in the first phase data points are only updated if the proposal point v_i improves on the function value of the original location x_i. Also in this phase the number of update attempts is kept track of.

The automated balancing of exploration on the one hand and exploitation on the other makes the Artificial Bee Colony algorithm a potentially strong optimisation algorithm. Moreover, it has the nice property that as the number of iterations increases, data points will tend to move towards minima, because updates are only performed if they improve the function value for the data point under consideration. This causes the updates of equation (2.17) to provide increasingly more resolution on these minima.

2.1.8 Gaussian Particle Filter

The Gaussian Particle Filter was first explored in ref. [131]. The scanning algorithm starts off by collecting an initial seed of randomly generated points. The number of points that are generated is a user definable quantity, as are the range and the sampling prior (uniform or logarithmic). By then using these randomly generated points as seeds, a multi-dimensional Gaussian distribution is used to draw new points around the location of those seed data points. The number of data points sampled from each Gaussian is proportional to the relative function value of the target function at the seed. The width of these Gaussians varies over the run time of the algorithm, starting at $(\mathcal{O}(1-2))$ at the start of the run, in order to cover a large part of the parameter space. In higher iterations, this width is shrunk by multiplying by a factor < 1. The speed at which it reduces is a hyperparameter of the algorithm and will henceforth be called the width decay.

The implementation of the algorithm that we employ in this paper[5] allows for the use of a uniform or logarithmic exploration, indicating if the standard deviations of the multi-dimensional Gaussian distributions should be multiplied by the coordinate x. This is configured through the boolean parameter `logarithmic`.

In each iteration N_{sample} data points are sampled from $N_{\text{Gaussians}}$ Gaussians and the function values for these samples are calculated. These samples are combined with the best fraction of the (already evaluated) samples that formed the seed for the Gaussians. This fraction is named the survival rate. From the resulting dataset, $N_{\text{Gaussians}}$ data points (i.e. those with the lowest function value) are selected as the seed for the Gaussians of the next iteration.

The algorithm is highly flexible, and is aided by adding user knowledge about how the function behaves. This can help to determine an appropriate value to choose for the width of the Gaussians. The stopping criterion of the implementation that we employed was based entirely on the width of the Gaussians, which we controlled using a width scheduling scheme. We did not use any information on the function values found, allowing the algorithm to potentially continue to sample after already finding the global minimum.

[5] https://github.com/bstienen/particlefilter.

JHEP05(2021)108

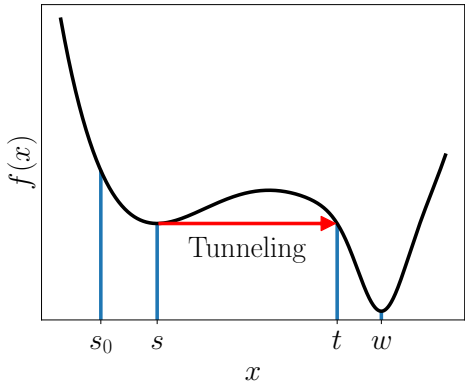


Figure 1. Visualisation of the tunneling approach for minimisation as used in AMPGO [132]. From the initial sample s_0 the local optimum at s is found. After it is found, Tabu Tunneling is used to find location t, from which the local gradient-based approach can be reused again to find the global minimum at w.

2.1.9 AMPGO

Adaptive Memory Programming for Global Optimisation (AMPGO) is a solver that uses multiple steps to solve a global optimisation problem [132]. First, a set of initial points in parameter space is chosen randomly. Next, a local solver is used to find the local optima of those points. When these are found, a method called Tabu Tunneling [133] is used to find a different local optimum in the parameter space from which the local optimiser is started again. Using this iterative approach, the global optimum can be found. In figure 1 a visualisation of this approach is shown. The goal is to find the global optimum w and the initial random point is s_0. As a first step, using the local optimiser, the point s is found. Then, using the Tabu Tunneling method, one tries to find a different point in parameter space with a value as good as s. In this example, that point is t. From t the local optimisation algorithm is invoked again to find the point w.

Any method can be used for the local solver; we employed L-BFGS-B [134]. A full mathematical description of the algorithm can be found in ref. [132]. The AMPGO website [135] contains benchmarks of 184 multidimensional test functions against multiple different algorithms. A Python implementation of AMPGO is available on GitHub.[6]

The algorithm relies heavily on tunneling towards a new region in parameter space with the same or lower function value. However, as the number of possible tunnelling directions from any point in parameter space is infinite, there are significant hurdles to overcome when trying to find a region with a lower or equal function value:

- When the global optimum is very narrow, the probability of tunnelling into the basin of attraction of this minimum can be very low (e.g. Analytic Function 1 discussed in section 3.1);
- When the dimensionality increases, the volume to tunnel through in order to find a new minimum grows exponentially;

[6] https://github.com/andyfaff/ampgo.

JHEP05(2021)108

- When there are many local minima of similar depth, the number of tunneling steps needed can be very high. Each time a new local minimum is found that is deeper than the last, the probability of finding a new minimum with an even lower function value goes down (see e.g. Analytic Function 2).

Given these difficulties, this algorithm can struggle to improve on its local solution in high-dimensional problems.

2.2 Characterisation of algorithms

It is not immediately obvious how to compare optimisation algorithms that depend on wildly different strategies, with each having a number of different hyperparameters. The best choice of the hyperparameters in each case will depend on the function being explored, and finding those best values is itself an optimisation challenge of reasonable complexity.

In this paper we attempt to provide the average particle astrophysicist with some knowledge of which algorithm might perform well for a particular type of function, without requiring them to engage in overly-aggressive hyperparameter optimisation. We therefore group the parameters of each of our techniques into four categories:

- **Convergence parameters.** These are parameters that affect the stopping point of an algorithm, or the point at which convergence is presumed to have occurred. A more stringent tolerance condition should have the effect of improving the best fit found by the algorithm, but will require a greater number of likelihood evaluations to reach that point.

- **Resolution parameters.** These are parameters that affect the resolution with which the target function is explored. A higher resolution will increase the detail with which the likelihood function is mapped around the best-fit point, leading to a better mapping of the 1σ, 2σ and 3σ confidence regions, at the cost of a greater number of likelihood evaluations in total. Higher resolution can also improve the final quality of the best-fit point, independently of convergence parameters.

- **Hint parameters.** These are parameters that give the algorithm a clue as to how to obtain the best solution. For example, algorithms that are required to start at a certain point would strongly benefit from starting near the global minimum of the function. A wise choice of such a parameter (if this is possible) would reduce the number of likelihood evaluations required to give a good fit.

- **Reliability parameters.** These are parameters whose general effect is to improve the robustness of a technique.

In table 1, we provide a grouping of the main parameters of each of our optimisation techniques, where it can be seen that not all techniques have a parameter in each group. Nevertheless, most of the techniques have a resolution and a convergence parameter, and the typical use-case for a particle astrophysicist would be to set these parameters to provide the best possible sampling within the available CPU budget, whilst using out-of-the-box

JHEP05(2021)108

Parameter	Explored values	Type
AMPGO		
Number of sampled points	2000, 5000, 10000, 20000	Resolution
CMA-ES		
Function tolerance	10^{-11}, 10^{-7}, 10^{-4}, 10^{-1}	Convergence
Population size (λ)	20, 50, 100, 500	Resolution
Diver		
Threshold for convergence	10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}	Convergence
Population size	2000, 5000, 10000, 20000	Resolution
Parameter adaptation scheme	λjDE	—
Gaussian Particle Filter		
Width decay	0.90, 0.95, 0.99	Convergence
Logarithmic sampling	`True`, `False`	Hint
Survival rate	0.2, 0.5	Reliability
Initial gaussian width	2	Reliability
GPyOpt		
Threshold for Convergence	10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}	Convergence
Particle Swarm Optimisation		
Threshold for convergence	10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}	Convergence
Population size	2000, 5000, 10000, 20000	Resolution
Adaptive ϕ	`True`	Reliability
Adaptive ω	`True`	Reliability
PyGMO Artificial Bee Colony		
Generations	100, 250, 500, 750	Resolution
Maximum number of tries	10, 50, 100	Reliability
PyGMO Differential Evolution		
Generations	100, 250, 500, 750	Resolution
Parameter adaptation scheme	iDE, jDE	—
PyGMO Grey Wolf Optimisation		
Generations	10, 50, 100, 1000	Resolution
random sampling		
Number of points	10, 50, 100, 500, 1000, 5000, 10000, 50000, 100000, 500000, 1000000	Resolution
Trust Region Bayesian Optimisation (TuRBO)		
Max #evaluations / iteration	64, 100	Convergence

Table 1. A grouping of the various parameters of each of our optimisation techniques into the categories described in the main text. The explored values for these parameters can be found in the second column.

choices for the hint and reliability parameters. Our results in section 4 will therefore be presented for different choices of the resolution and convergence parameters for each algorithm (where these are available).

JHEP05(2021)108

3 Definition of sampler comparison tests

3.1 Analytic test functions

In this paper two sets of tests are performed. In the first, we compare the performance of each of the above algorithms on a series of analytic test functions, each of which embodies a different pathology that is commonly asociated with difficult physics examples. In the second, a real physics example is used, in order to determine whether the insights gained on analytic test functions generalise to a more realistic setting.

We use four reference functions[7] to evaluate and compare the performance of every optimisation algorithm. The analytic forms of these functions were unknown to the operators of the optimisation algorithms until the results were generated, in order not to introduce biases towards particular global minima.

Analytic function 1. The equation for Analytic Function 1 reads as follows:

$$f(\mathbf{x}) = e^{-\sum_{i=1}^{n}((x_i-2)/15)^6} - 2e^{-\sum_{i=1}^{n}(x_i-2)^2} \prod_{i=1}^{n} \cos^2(x_i - 2), \tag{3.1}$$

where n is the number of dimensions. This function is visualised in figure 2a for 2-dimensional input. The global minimum is at $\mathbf{2}$, where a function value of -1 is reached. This function is expected to be difficult to optimise numerically, as this minimum is surrounded by a region with relatively high function values. The global minimum is put at $\mathbf{2}$ rather than $\mathbf{0}$ to discourage algorithms that take zero as a starting point. The minimum of this function is searched for in the domain $[-30.0, 30.0]^n$.

Analytic function 2. The equation for Analytic Function 2 reads as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left[(x_i + 0.23)^2 - 10\cos\left(2\pi(x_i + 0.23)\right) + 10 \right], \tag{3.2}$$

where n is the number of dimensions. This equation is visualised in figure 2b for 2-dimensional input. The difficulty in this analytic function is that there are many local minima that the optimisation algorithm can get stuck in. The global minimum is put at $-\mathbf{0.23}$ rather than $\mathbf{0}$ to discourage algorithms that take zero as a starting point, with $f(-\mathbf{0.23}) = 0$. The minimum of this function is searched for in the domain $[-7.0, 7.0]^n$.

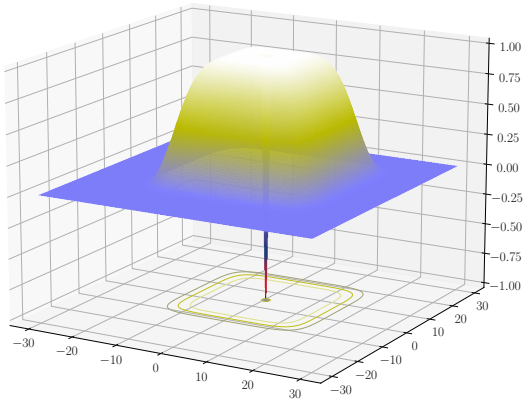
Analytic function 3. The equation for Analytic Function 3 reads as follows:

$$f(\mathbf{x}) = -\frac{1}{n} \sum_{i=1}^{n} \sin^6 \left[5\pi \left(x_i^{3/4} - 0.05 \right) \right], \tag{3.3}$$

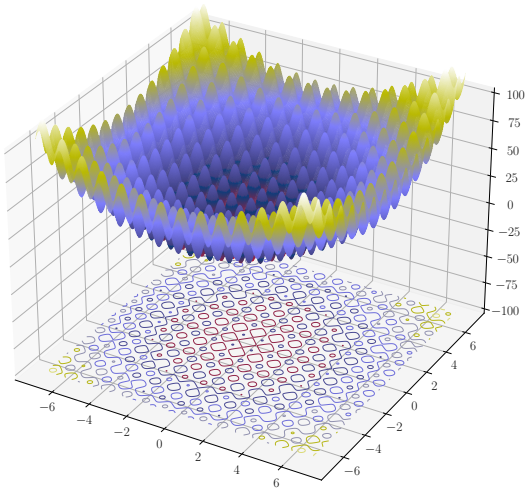
where n is the number of dimensions. This equation is visualised in figure 2c for 2-dimensional input. This test function has many global minima and the optimisers should be able to find the global minimum, which is at -1. The minimum of this function is searched for in the domain $[0.0, 1.0]^n$.

[7] Our choice of these functions is greatly influenced by [135].

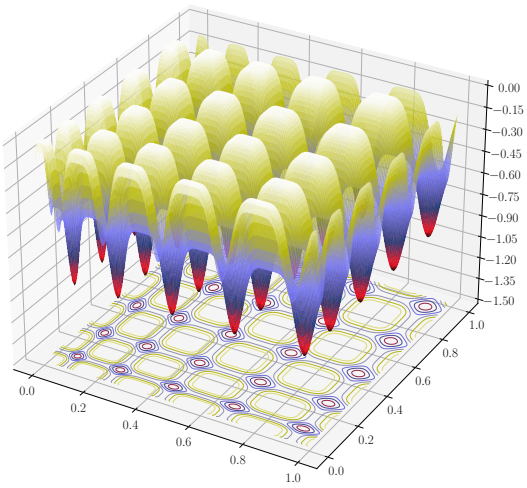
JHEP05(2021)108



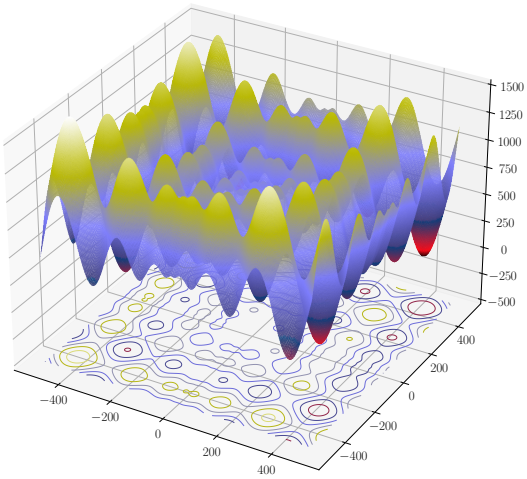
(a) Analytic Function 1 (eq. (3.1)) — global minimum at **2**.



(b) Analytic Function 2 (eq. (3.2)) — global minimum at $-$**0.23**.



(c) Analytic Function 3 (eq. (3.3)) — many degenerate minima.



(d) Analytic Function 4 (eq. (3.4)) — global minimum at about **421**.

Figure 2. Visualisation of the explored analytic functions from section 3.1 in 2-dimensional form.

Analytic function 4. The equation for Analytic Function 4 reads as follows:

$$f(\mathbf{x}) = 418.9829n - \sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right), \tag{3.4}$$

where n is the number of dimensions. This equation is visualised in figure 2d for 2-dimensional input. The difficulty in this analytic function is that it has an irregular shape and the parameter range is quite large. The global optimum is at about **420.968746** with $f($**420.968746**$) \simeq 0$. The minimum of this function is searched for in the domain $[-500.0, 500.0]^n$.

JHEP05(2021)108

3.2 Particle astrophysics test problem

In addition to the test functions described above, it is interesting to test our various optimisation algorithms on a realistic particle astrophysics problem. A leading use of optimisation techniques in particle astrophysics is in global fits of models beyond the Standard Model of particle physics. These add a number of additional parameters to the Standard Model, and one must find the regions of the extended parameter space that are most compatible with current experimental data. In frequentist statistics, this is typically performed by maximising a likelihood function $\mathcal{L}$, which is equivalent to minimising $-\log \mathcal{L}$. This problem therefore resembles the problem of minimising the analytic functions.

For our example, we take a recent global fit of a supersymmetric theory performed by the GAMBIT collaboration in ref. [103], and obtain a fast interpolation of the likelihood function that was originally computationally expensive to obtain. The original fit explored a 7-parameter phenomenological version of the Minimal Supersymmetric Standard Model (the so-called "MSSM7"), which is described by the soft masses M_2, $m^2_{\tilde{f}}$, $m^2_{H_u}$, $m^2_{H_d}$, the trilinear couplings for the third generation of quarks A_{u_3}, A_{d_3} and $\tan\beta$ (plus the input scale $Q = 1\,\text{TeV}$ and the sign of μ, which was chosen to be positive). The mass parameters above are all defined at the Q common scale whereas $\tan\beta$ is defined at m_Z. In addition to these supersymmetric model parameters, the original fit added a variety of nuisance parameters, comprising the strong coupling constant, the top quark mass, the local dark matter density, and the nuclear matrix elements for the strange, up and down quarks. Therefore, the global fit was performed on a 12-dimensional parameter space.

To make it possible to compare the performance of the different algorithms considered in this work we have approximated the joint likelihood[8] using a deep neural network as proposed e.g. in [136, 137]. The total number of samples collected from the global fit to train the network was about 2.3×10^7. The network consists out of four hidden layers of 20 fully-connected neurons each, activated through the SELU function [138]. We have normalized the input and output data to a normal Gaussian distribution and split the data into 90% for training and 10% for testing. We have set the batch size for training to 1024 and optimised the performance by halving the learning rate of the Adam optimiser [139] when the loss function, in this case the mean absolute error (MAE), stopped improving. Early stopping was applied to stop training after a couple of these iterations.

Figure 3 shows the validation plot of the trained network, where it can be seen that the neural net prediction of the likelihood is well-correlated with the true likelihood. It is important to note that perfect performance of the fast likelihood interpolation is not required, as for our purposes it is sufficient that it provides a suitable proxy for a difficult likelihood function that would typically be encountered in a particle astrophysics application.

4 Results

In this section, we compare the results of running the optimisation methods described in section 2 on the functions described in section 3. To make sure the algorithm was the

[8] See table 3 of ref. [103] for details about which data were used to define it.

JHEP05(2021)108

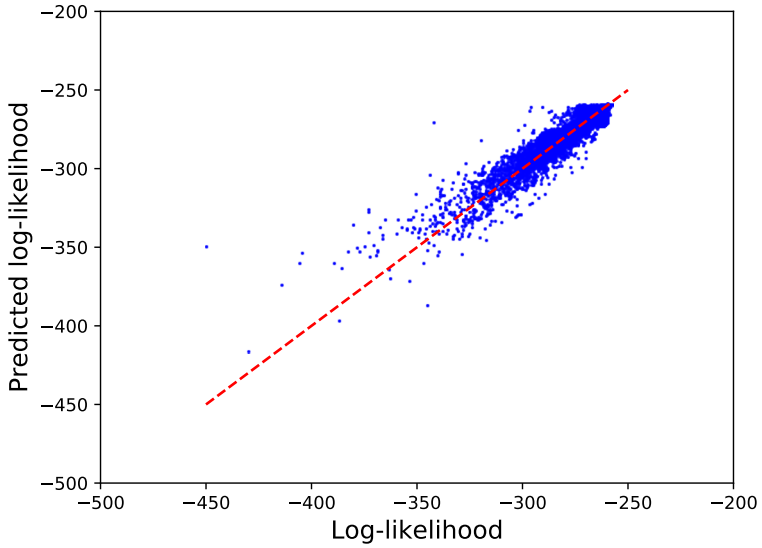


Figure 3. Validation plot of the trained network, showing the true log likelihood on the x-axis and the predicted value on the y-axis.

only difference between all these optimisation experiments (apart from the algorithms' hyperparameters), we performed all experiments using the same Python framework. A description of this framework can be found in appendix A. A comprehensive overview of the best found result for each of the algorithms on each optimised function (for all explored dimensionalities) can be found in appendix B.

4.1 Analytic functions

We first present results for the analytic test functions described in section 3.1. These four functions were explored as 2-dimensional, 3-dimensional, 5-dimensional and 7-dimensional functions. It is expected that the ability of optimisation algorithms to find the true minimum decreases as the dimensionality increases. Each algorithm was run for several sets of its resolution and convergence hyperparameters, which are summarised in table 1.

Figures 4 to 7 show the accuracy with which each algorithm recovered the minimum of each function for each of the dimensionalities, with each circle representing a particular run of each algorithm with a specific choice of hyperparameters. The size of the circles is proportional to $\log_{10}$ of the total number of likelihood evaluations in that run, whilst the different lines for each algorithm correspond to different dimensionalities (going from 2D for the line on the bottom, to 7D for the top-most line). To illustrate the accuracy, we show the difference between the known global minimum and the best value found for each run.

The results for Analytic Function 1 show that some algorithms never get anywhere close to the global minimum, regardless of the dimensionality of the problem. In ≥ 3 dimensions, all algorithms fail to find the global minimum. Inspecting the function in 2D (see figure 2a) reveals why — there is a very spiked local minimum that is hard to locate, a problem that will get even harder as the dimensionality of the function increases. The

JHEP05(2021)108

best-performing algorithm, such as it is, is the PyGMO Artificial Bee Colony, which finds the correct minimum in 2D and 3D with a relatively small number of likelihood evaluations. The worst performing algorithms are PyGMO Grey Wolf Optimisation, Gaussian Particle Filter, AMPGO, GPyOpt and the PyGMO implementation of Differential Evolution. In the latter case this may simply be due to the low number of total likelihood evaluations, suggesting that a more stringent set of hyperparameters might yield better performance. This is confirmed by the fact that the Diver performance is apparently better, giving the correct global minimum in 2D with more likelihood evaluations and better, though not adequate, performance in 3D. The Gaussian Particle Filter algorithm works in 2D, but this is almost certainly due to the fact that it has performed a large number of likelihood evaluations in a low-dimensional space. It is outperformed by random sampling in 3D, as are all algorithms except PyGMO Artificial Bee Colony and TuRBO. Finally, it is worth comparing the performance of the two Bayesian Optimisation algorithms. The failure of GPyOpt to find the global minimum in any dimensionality is not unexpected, as a sharply-spiked local minimum is exactly the case that is expected to be missed by Bayesian Optimisation due to its relatively low number of samples of the objective function, and concentration of those samples in areas where the algorithm thinks it has found interesting points. TuRBO, meanwhile, is able to find the global minimum correctly in 2D (and reasonably well in 3D) because it breaks up the space into separate regions, and runs an independent Bayesian Optimisation within each of them. One of these regions is small enough to contain the global minimum as the obvious minimum, rather than the plateau of false minima at the edge of the function range.

For the second analytic function, it is interesting to see that most algorithms are able to find the global minimum. Moreover, the fact that most of these algorithms are able to systematically outperform random sampling indicates that these algorithms are in fact effective methods for optimisation problems that resemble Analytic Function 2, with a large number of local minima but a single clear global minimum. AMPGO is the only algorithm that fails in all numbers of dimensions, being outperformed by random sampling, followed by GPyOpt, which only properly succeeds in 2D. TuRBO performs better, giving adequate performance in 2D and 3D, but at the cost of a large number of total likelihood evaluations. The best algorithm is the PyGMO implementation of Differential Evolution, which gets the correct answer in all dimensionalities with a low number of likelihood evaluations. Diver also performs well, with a higher number of evaluations, but the PyGMO results suggest that fewer Diver evaluations would still give good performance. It is interesting to note that the PyGMO Artificial Bee Colony still performs well, getting the correct answer in all dimensionalities with a relatively low number of evaluations. A final interesting feature of the results in figure 5 is that — as expected — the performance of each algorithm deteriorates with increasing dimensionality, as can be seen from the increased spread of results for higher dimensionalities.

The results for Analytic Function 3 in figure 6 show that AMPGO retains its status as the worst algorithm, once again being outperformed by random sampling. However, in 2D the global minimum is almost found, with a fairly modest number of likelihood evaluations. In general, all of the algorithms now show very good performance, which is a reflection

JHEP05(2021)108

of the fact that there are now many global minima in the function, and it is easy to find at least one of them. This is further reflected in the fact that the precise configuration of each algorithm now becomes less important, with much less variation in the results of runs with different hyperparameter settings. Indeed, this is the only example among the analytic functions where a local optimisation approach would work. GPyOpt now shines, and finds the global minimum correctly in all dimensionalities with the smallest number of total likelihood evaluations. PyGMO Artificial Bee Colony still does a good job, with a relatively small number of likelihood evaluations. Comparing the Differential Evolution implementations we see that Diver does not quite get the global minimum in 7D, whilst the PyGMO Differential Evolution implementation finds it in all cases, with fewer likelihood evaluations. TuRBO is able to match the performance of GPyOpt, but with more likelihood evaluations due to the requirement of running many separate optimisations (these extra optimisations are redundant in this case).

Analytic Function 4 has various local minima, but only one global minimum. In figure 7, we see that random sampling outperforms AMPGO again. PyGMO Artificial Bee Colony gets the answer right in all dimensionalities, as do the two Differential Evolution algorithms, CMA-ES, and Particle Swarm Optimisation. The best performing algorithm is now the PyGMO implementation of Differential Evolution, although again we should caution that Diver may give similar performance for a suitable choice of its hyperparameters. Again we see Bayesian Optimisation fail as the dimensionality increases, although TuRBO is better than GPyOpt in >5 dimensions. Of particular note is the fact that the Grey Wolf Optimisation now does not perform well at all, and seems only to have worked for Analytic Function 2 and Analytic Function 3.

In summarising the performance of the 11 different algorithms on the four different analytic functions, we can ask whether any algorithm emerges with acceptable performance on all of them. A summary of their performance is given in table 2. PyGMO Artificial Bee Colony emerges as perhaps the best candidate, since it performs well for all functions except Analytic Function 1, and it gives the best performance in that case. AMPGO is poor and consistently worse than random sampling. The success or failure of Bayesian Optimisation (GPyOpt and TuRBO) is interesting to investigate across the analytic functions. It generally fails when there are hidden minima, but the situation can be improved by adding latin hypercube sampling such as that found in the TuRBO algorithm. Differential Evolution is consistently strong in both of the PyGMO and Diver implementations, whilst CMA-ES also shows consistent performance (all except for Analytic Function 1). Particle Swarm Optimisation is not as consistent across the analytic functions, and where it succeeds it requires a large number of evaluations. Finally, the Gaussian Particle Filter algorithm struggles in higher dimensionalities, and is highly sensitive to details of its configuration.

In appendix B, table 3 to table 6, we show the best minimum found by each algorithm for each analytic function and dimensionality. The hyperparameter settings shown in each case are those that led to the best minimum and, where multiple different runs obtained the same result, the settings are those that needed the smallest number of total likelihood evaluations to obtain that result.

JHEP05(2021)108

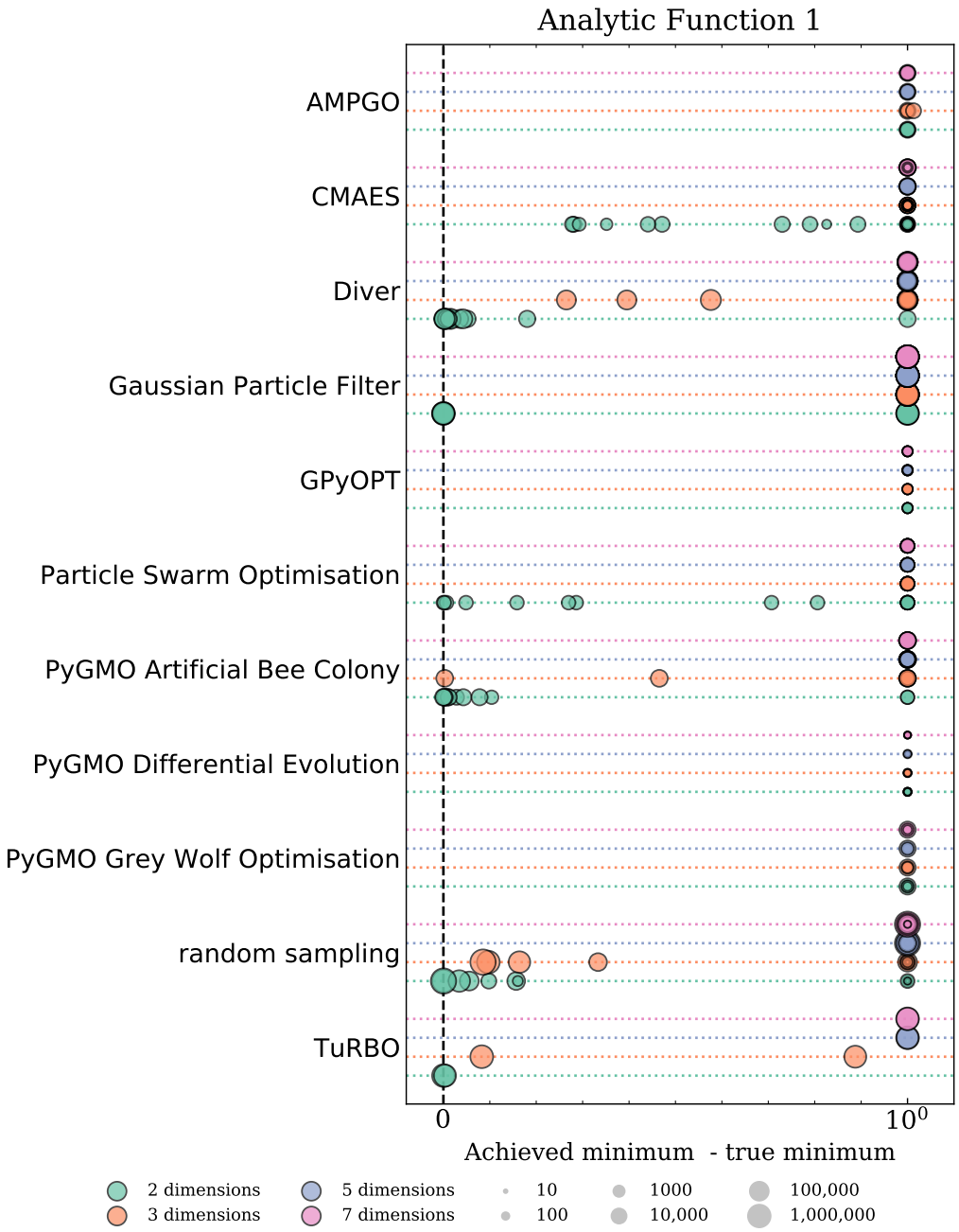


Figure 4. Results from different optimisation algorithms on the analytic function in equation (3.1). The results are shown as semi-opaque circles, of which the area increases logarithmically with the number of function evaluations needed to obtain that specific result. The four horizontal lines for each algorithm belong to the four explored dimensionalities, from top to bottom 7-dimensional (pink), 5-dimensional (purple), 3-dimensional (orange) and 2-dimensional (green). The horizontal axis shows the difference between the (known) log-likelihood at the global minimum and that at the found minimum.

JHEP05(2021)108

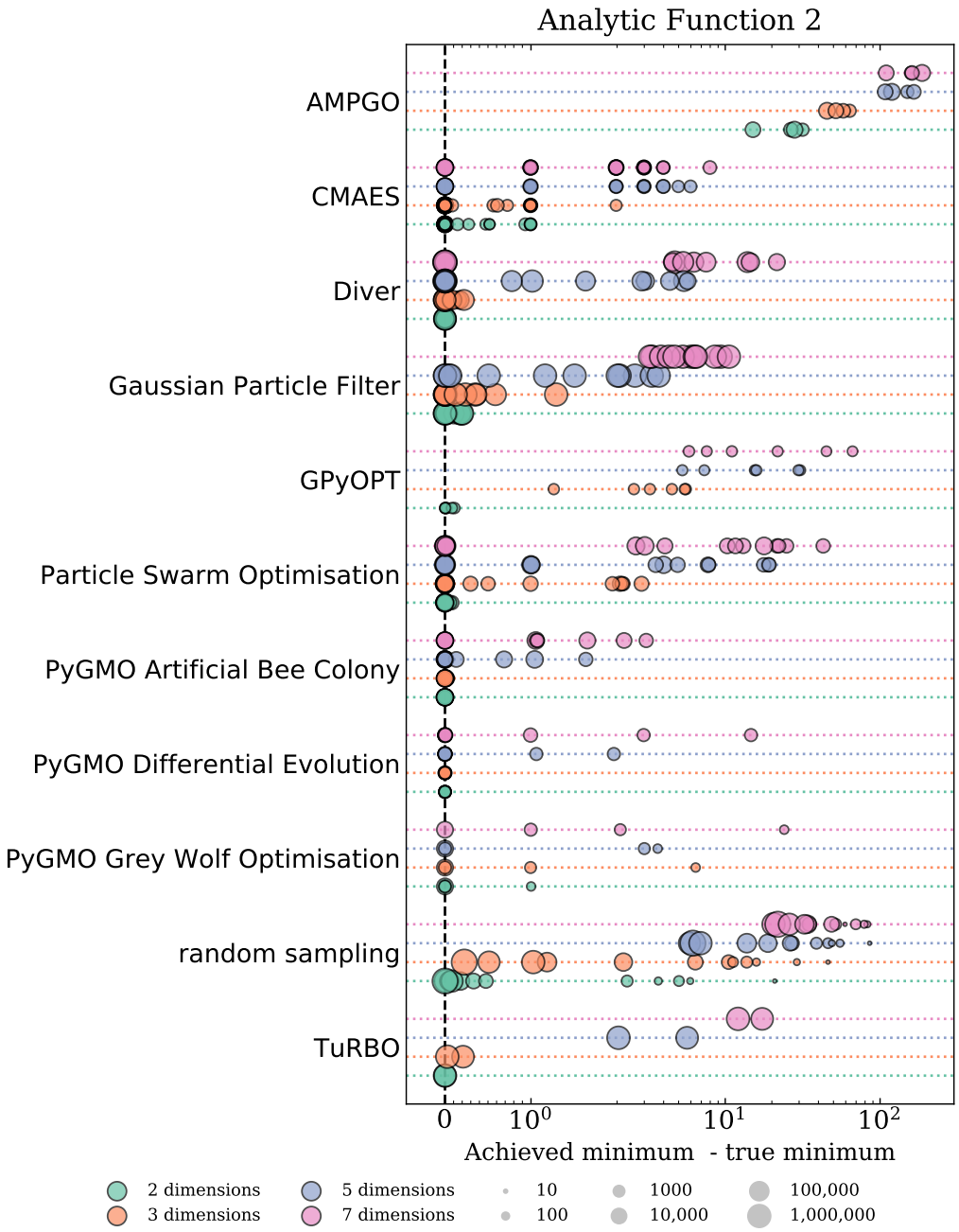


Figure 5. Results from different optimisation algorithms on the analytic function in equation (3.2). The results are shown as semi-opaque circles, of which the area increases logarithmically with the number of function evaluations needed to obtain that specific result. The four horizontal lines for each algorithm belong to the four explored dimensionalities, from top to bottom 7-dimensional (pink), 5-dimensional (purple), 3-dimensional (orange) and 2-dimensional (green). The horizontal axis shows the difference between the (known) log-likelihood at the global minimum and that at the found minimum.

JHEP05(2021)108

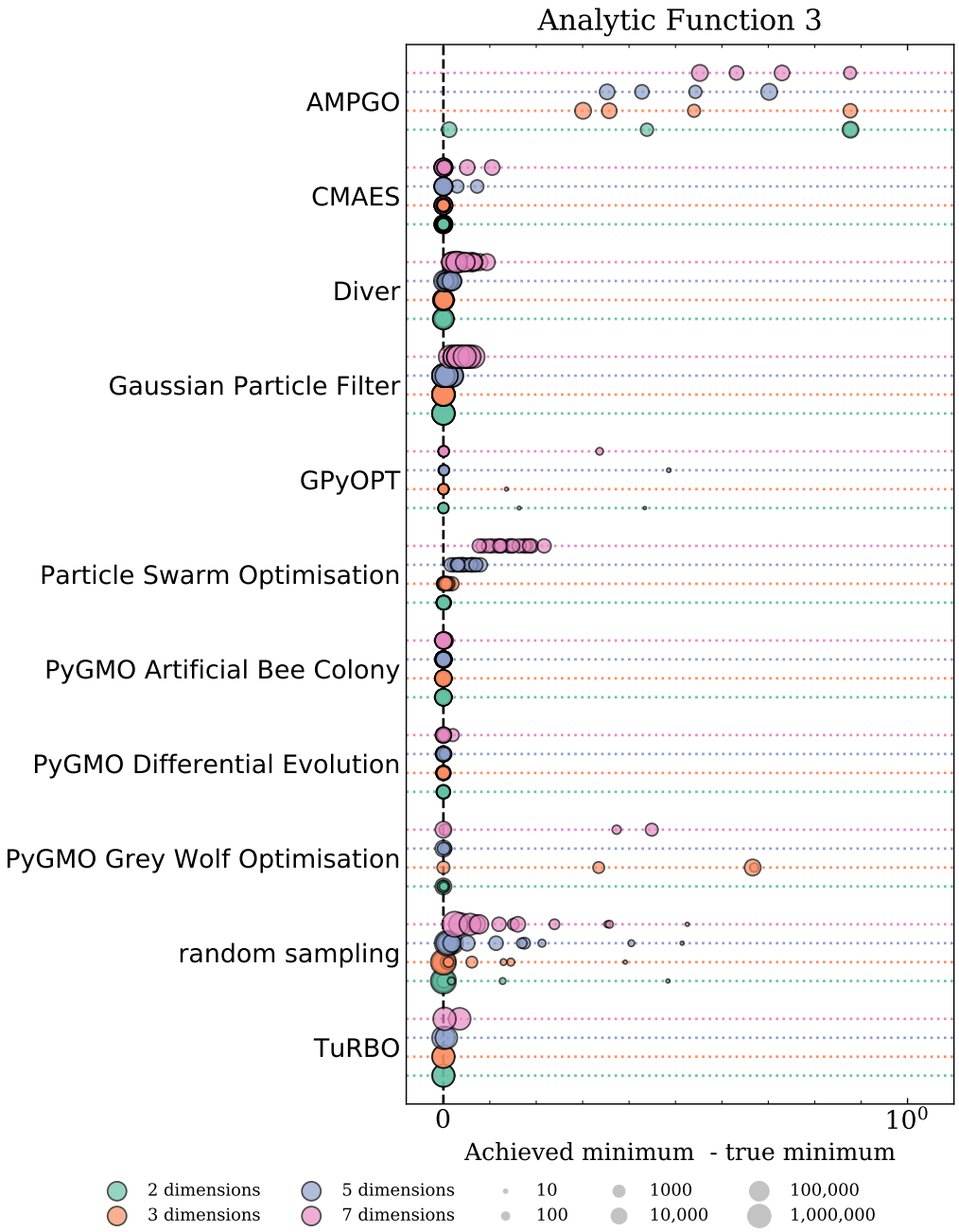


Figure 6. Results from different optimisation algorithms on the analytic function in equation (3.3). The results are shown as semi-opaque circles, of which the area increases logarithmically with the number of function evaluations needed to obtain that specific result. The four horizontal lines for each algorithm belong to the four explored dimensionalities, from top to bottom 7-dimensional (pink), 5-dimensional (purple), 3-dimensional (orange) and 2-dimensional (green). The horizontal axis shows the difference between the (known) log-likelihood at the global minimum and that at the found minimum.

JHEP05(2021)108

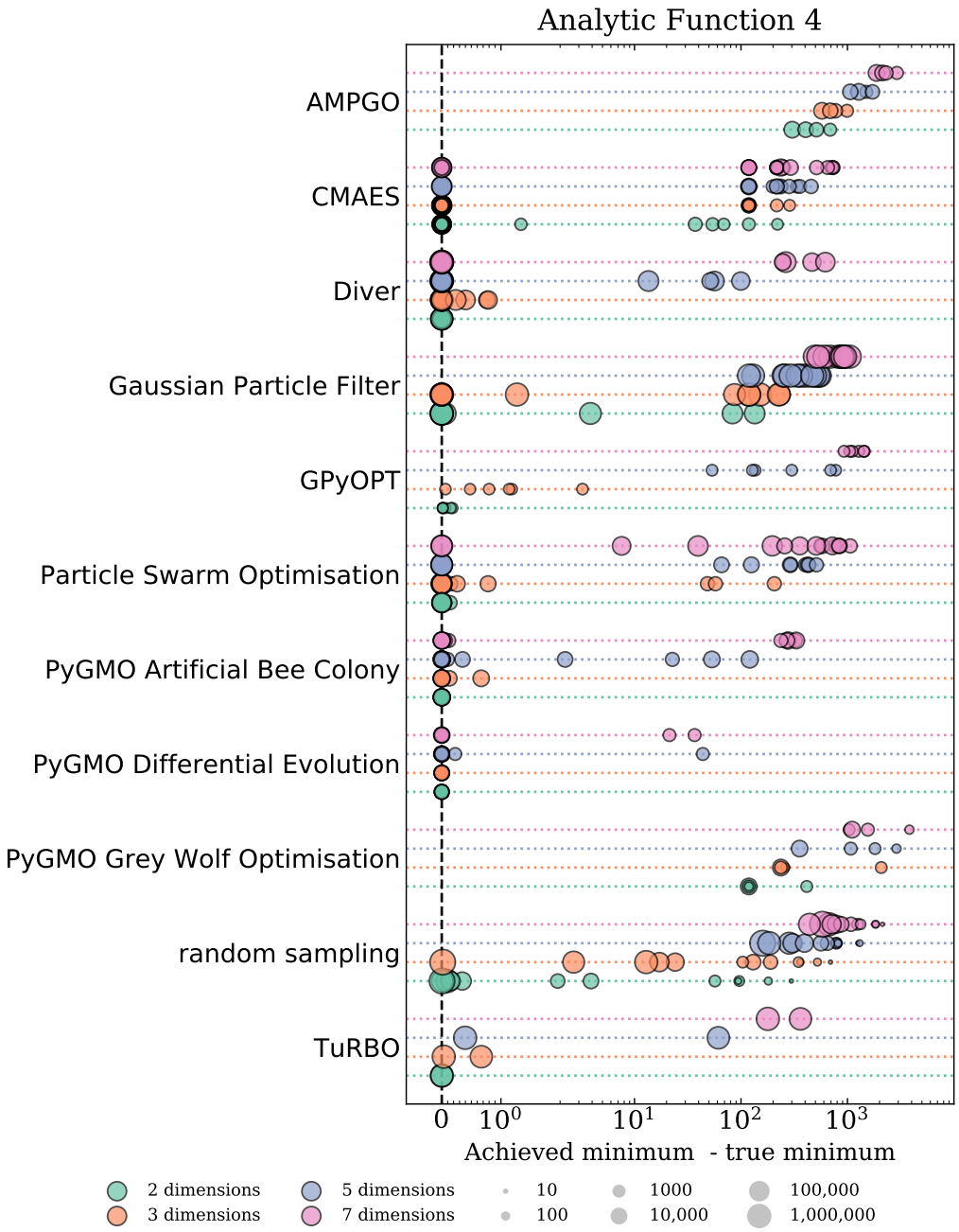


Figure 7. Results from different optimisation algorithms on the analytic function in equation (3.4). The results are shown as semi-opaque circles, of which the area increases logarithmically with the number of function evaluations needed to obtain that specific result. The four horizontal lines for each algorithm belong to the four explored dimensionalities, from top to bottom 7-dimensional (pink), 5-dimensional (purple), 3-dimensional (orange) and 2-dimensional (green). The horizontal axis shows the difference between the (known) log-likelihood at the global minimum and that at the found minimum.

JHEP05(2021)108

4.2 Particle astrophysics test problem

In figure 8, we show the results for each algorithm for the MSSM7 test example described in section 3.2. The immediate thing to note is that Diver emerges as the best algorithm, finding the best fit of all algorithms. It comfortably outperforms the PyGMO implementations of Differential Evolution, albeit with a higher number of likelihood evaluations (suggesting, once more, that the PyGMO code may give better results for different choices of the hyperparameters). The cause for this exceptionally strong performance of Diver might be the fact that the training data for the neural network was itself sampled by Diver. Although the training data was created independently of any of the optimisation experiments presented here, a neural network trained on that data might still encode the patterns typically explored by Diver, while not encoding the patterns used by the other algorithms equally well. However, the neural network still provides an example of a physically-motivated function and it is true to say that the non-Diver algorithms were not able to find the minimum of this function.

Apart from Diver, many of the algorithms give similar performance, comparing favourably to random sampling. Although random sampling is, for this physics test problem, able to come as close to the same solution as many other algorithms, it needs significantly more function evaluations to achieve this performance. The PyGMO Artificial Bee Colony algorithm is, in a surprising turn of events, not notably better than most of the other algorithms, even though it performed consistently well on the analytic functions.

The two Bayesian Optimisation methods, GPyOpt and TuRBO, are amongst the algorithms that perform better than, or comparable to, random sampling, but they underperform relative to other algorithms in this group. This is likely caused by the dimensionality of the problem (12D); we already saw in the results of the analytic functions that an increase in dimensionality dragged the performance of these algorithms down strongly.

The only exception to the general trend of "at least similar performance to random sampling" is AMPGO, which remains consistently poor.

5 Conclusions

We have performed a detailed comparison of a variety of optimisation algorithms in an attempt to find new algorithms for particle astrophysics problems. Many of these algorithms have not been used in a particle astrophysics context before, and we have examined their ability to find the correct global minimum of a range of test functions. They were also tested in their ability to correctly maximise a likelihood in a realistic particle astrophysics example based on a recent global fit of a phenomenological supersymmetry model. The algorithms we investigated were Differential Evolution (using two different software implementations), Particle Swarm Optimisation, the Covariance Matrix Adaptation Evolution Strategy, Bayesian Optimisation (in two different forms), Grey Wolf Optimisation, PyGMO Artificial Bee Colony, Gaussian Particle Filter and AMPGO. All of the algorithms used in our comparison have publicly-available software implementations.

For each algorithm, we characterised the hyperparameters as affecting the convergence or resolution of the optimisation, or as providing hints or improving the reliability. We

JHEP05(2021)108

	Finding sharp minimum	Finding global minimum	Performance with high dimensions	Average number of evaluations
AMPGO	bad	bad	bad	low
CMA-ES	very low dimensions	good	good	medium
Diver	low dimensions	good	good	high
Gaussian Particle Filter	very low dimensions	low dimensions	highly configuration and function dependent	high
GPyOpt	bad	low dimensions	function dependent	low
Particle Swarm Optimisation	very low dimensions	good	configuration dependent	medium
PyGMO Artificial Bee Colony	low dimensions	good	good	medium
PyGMO Differential Evolution	bad	good	good	low
PyGMO Grey Wolf Optimisation	bad	bad	function dependent	medium
TuRBO	low dimensions	moderate dimensions	function dependent	high
random sampling	low dimensions	low dimensions	function dependent	high

Table 2. Summary of optimisation algorithm performance. Analytic Function 1 can be used to evaluate the performance of algorithms for finding a sharp minimum. Analytic Function 4 can be used to compare the performance of algorithms for finding a global minimum. The performance of algorithms for increasing number of dimensions and average number of evaluations was compared for all four analytic functions.

JHEP05(2021)108

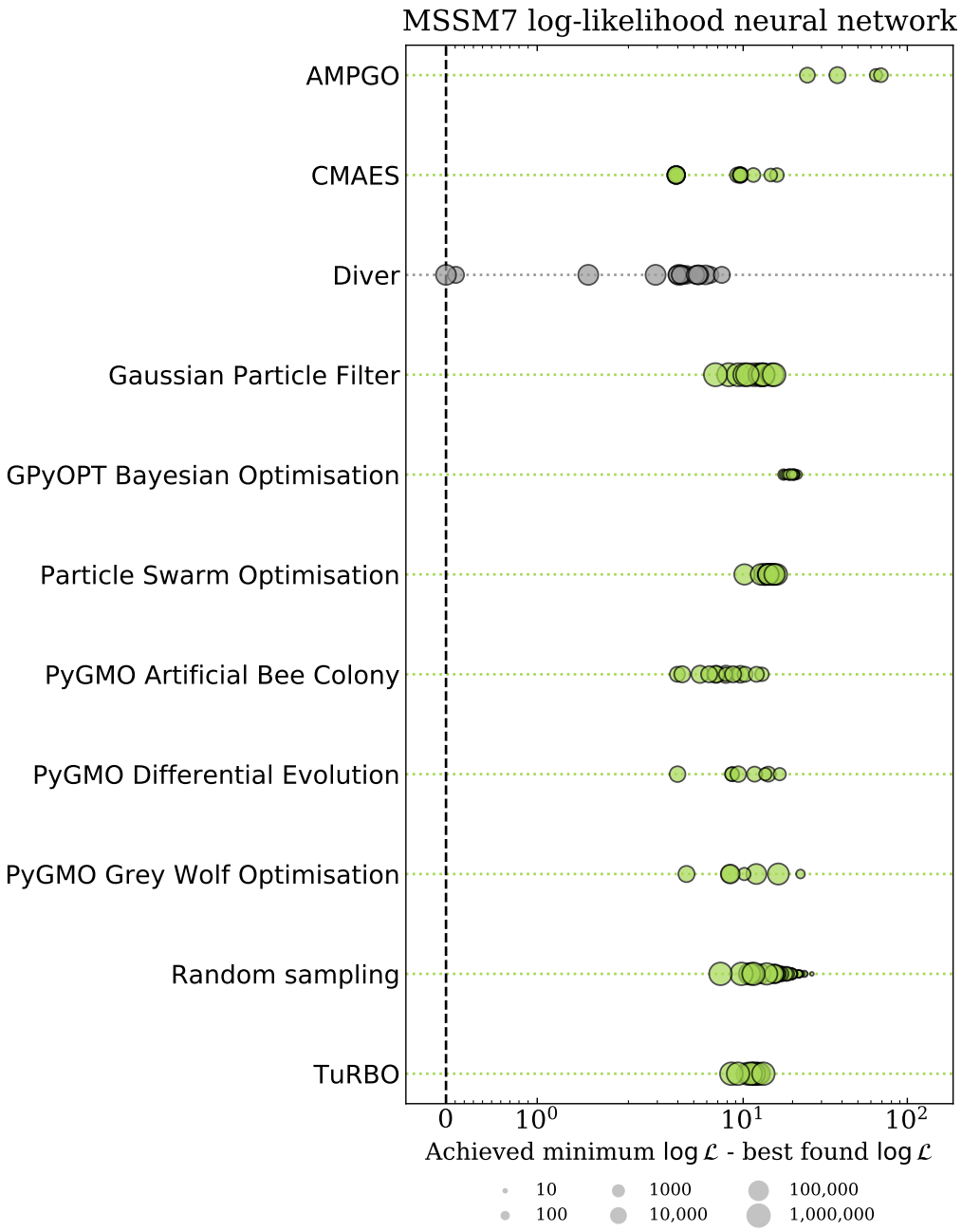


Figure 8. Results from different optimisation algorithms on the neural network approximation of the MSSM7 log-likelihood described in section 3.2. The results are shown as semi-opaque circles, of which the area increases logarithmically with the number of function evaluations needed to obtain that specific result. The horizontal axis shows the difference between the log-likelihood at the found minimum and the deepest minimum found by any algorithm for any settings. To emphasise the possible bias in the test function towards Diver, the results for that algorithm are coloured differently.

JHEP05(2021)108

then ran the different algorithms with different resolution and convergence hyperparameter settings, and compared the performance on test functions of different dimensionality, and our custom implementation of the MSSM7 likelihood function. Understandably, our main conclusion is the almost facile observation that the "best" algorithm depends strongly on the type of function that one wishes to optimise. However, it is possible to add some further interesting conclusions:

- Algorithms that emerge as the most consistent performers when evaluated on analytic functions do not necessarily give the best performance on a realistic particle astrophysics example. This is evidenced by the fact that the PyGMO Artificial Bee Colony algorithm arguably emerged as the most consistent performer on our analytic functions. It struggled to match the performance of the Differential Evolution implementation Diver on the MSSM7 likelihood function, but this might be due to a bias towards Diver in this physics inspired test case.

- Differential evolution (in various implementations) performed consistently well across the full barrage of tests.

- AMPGO performed consistently poorly on all test examples, being outperformed by random sampling in most cases.

- Bayesian Optimisation (in two variants, standard Gaussian Process-based Bayesian Optimisation, implemented in GPyOpt, and the Trust Region Bayesian Optimisation algorithm, implemented in the TuRBO package) performs well for functions with many global minima, but struggles in cases with very sharply-peaked minima, or multiple global minima. Performance can be enhanced by performing separate optimisations in different latin hypercubes, at the cost of increasing the total number of likelihood evaluations. Even then, the performance degraded significantly for the analytic functions once the dimensionality increased.

Finally, many of the algorithms used here show promising performance both on the analytic functions and the particle astrophysics example. This certainly motivates their use in real-world particle astrophysics applications, and we look forward to reading future examples of their application.

A Description of DarkMachines sampling framework

All experiments run for this paper were performed in an open-source Python package written specifically for this research: the High-Dimensional Sampling framework, henceforth abbreviated to HDS. This appendix outlines the general workings of, and the design ideas behind this package. It is however by no means a manual. A more complete and more technical introduction to the package can be found on the wiki at the project GitHub page: https://github.com/DarkMachines/high-dimensional-sampling/. The full code is published under the MIT license.

JHEP05(2021)108

Design considerations. As many different algorithms needed to be tested for this paper, a consistent test framework was needed. Ideally this meant that this framework would be invariant under changes of the optimisation algorithm used for investigation: it should just use the supplied optimisation method and test its performance just like it would test the performance of any other optimisation method when supplied. As the intention was to use a wide range of optimisation methods — of which a significant number were already implemented in packages like ScannerBit [111] — no single existing package was found that perfectly matched our requirements.

In designing and writing the custom package the following guidelines were therefore followed:

1. The package should make experiments reproducible;
2. The evaluation of the performance of an optimisation algorithm should not depend on the exact algorithm under investigation;
3. The package should automate as much of the experiments as possible, with a minimum loss of configurability;
4. The package should be easy to use and install, as experiments will be performed by many people on many different machines;
5. The output of the package should make it possible to easily compare the performance of different algorithms.

Core components. Following guideline (2) the package does indeed not depend on any optimisation algorithm: experiments can run with any implemented algorithm and these can be freely interchanged. However, to achieve this independence a common interface is needed. In the HDS framework this interface is implemented as the `Procedure` class. This class implements a small selection of functions that give other parts of the framework the possibility to query the procedure for new samples or for its status (e.g., whether or not it has finished sampling and whether or not it can optimise a specific target function). As the interface is very minimal (there are only 5 functions that need to be implemented), implementing new optimisation methods is relatively easy. Implemented Procedures can be found in the `optimisation` submodule of the package. As some of these require the installation of extra third-party packages, it is recommended to read the documentation on the GitHub wiki for more information.

Optimisation procedures are tested on test functions. To open up the possibility of having third-party test functions (i.e., functions defined by an external likelihood evaluation procedure), these functions also have an interface class: `TestFunction`. The majority of the `TestFunction`s have their analytic form programmed directly in the python code. They include common optimisation targets, like the Himmelblau function, the Rastrigin function and the ThreeHumpCamel function (see the GitHub wiki for a complete list of implemented functions and references to their analytic forms).

Although useful as tests of optimisation procedures, the fact that the user has access to the analytic form of these functions (or can just google for the coordinate and functional

JHEP05(2021)108

value of the optimum), they are not true blind tests of optimisation procedures. Because of this, four additional functions are implemented for which the pre-compiled binaries are included in the package. The analytic forms of these functions are not included in the package. These so-called `HiddenFunction`s are the functions referred to in section 3.1.

Experiments are run using instances of the `Experiment` class. It is this class that guarantees the first and third design guidelines. Providing the configured optimisation `Procedure` and defining on which `TestFunction`s the procedure should be tested, the `Experiment` class runs the experiment and outputs all necessary information to interpret the procedure's performance. This information includes:

- Benchmarks of the speed of the machine on which the computer is run;
- For each used `TestFunction`:
  - Meta data about the Procedure and the TestFunction (e.g., values of the configurable parameters);
  - The number of times the function was called (and if applicable the number of times the function was queried for its derivative);
  - The coordinates of the taken samples;
  - The found optimum and the function value at that coordinate;

B Best found results and parameter settings

Tables 3 to 7 show for each explored optimisation algorithm the configuration and result of the run which came closest to the global minimum (for the analytic functions) or the overall best found minimum (for the MSSM7 function). If multiple runs resulted in the same function value, the result with the least number of function evaluations is shown.

JHEP05(2021)108

Algorithm	dim	Parameters	min	N_{eval}
AMPGO	2		-0.0	5013
	3		-0.0	20016
	5		-0.0	10008
	7		-0.0	20120
CMA-ES	2	convergence=1.0000000000000001e-11, resolution=100.0	-0.722	10900
	3	convergence=0.1, resolution=20.0	0.0	160
	5	convergence=1.0000000000000001e-11, resolution=20.0	0.0	20
	7	convergence=0.1, resolution=20.0	0.0	20
Diver	2	convthresh=0.0001, np=5000	-0.998	65000
	3	convthresh=0.1, np=10000	-0.735	110000
	5	convthresh=0.001, np=2000	0.0	22000
	7	convthresh=0.1, np=2000	0.0	22000
Gaussian Particle Filter	2	logaritmic=True, survival_rate=0.2, width_decay=0.9	-1.0	225589
	3	logaritmic=False, survival_rate=0.5, width_decay=0.9	0.0	469469
	5	logaritmic=True, survival_rate=0.5, width_decay=0.95	0.0	983262
	7	logaritmic=True, survival_rate=0.2, width_decay=0.95	0.0	977481
GPyOpt	2	eps=0.1	0.0	511
	3	eps=0.0001	0.0	425
	5	eps=0.01	0.0	345
	7	eps=0.001	0.0	456
Particle Swarm Optimisation	2	convthresh=0.001, np=10000	-1.0	4400
	3	convthresh=0.1, np=20000	0.0	4000
	5	convthresh=0.1, np=20000	0.0	4000
	7	convthresh=0.1, np=10000	0.0	4000
PyGMO Artificial Bee Colony	2	generations=750, limit=50	-1.0	30020
	3	generations=750, limit=100	-0.997	30020
	5	generations=100, limit=50	0.0	4020
	7	generations=100, limit=10	0.0	4020
PyGMO Differential Evolution	2	generations=500, variant=iDE	0.0	80
	3	generations=750, variant=jDE	0.0	60
	5	generations=250, variant=iDE	0.0	40
	7	generations=750, variant=jDE	0.0	40
PyGMO Grey Wolf Optimisation	2	generations=10	0.0	220
	3	generations=10	0.0	220
	5	generations=10	0.0	220
	7	generations=10	0.0	220
random sampling	2	n_samples=1000000	-1.0	1000000
	3	n_samples=1000000	-0.903	1000000
	5	n_samples=10	0.0	10
	7	n_samples=10	0.0	10
TuRBO	2	max_eval=100	-1.0	1001876
	3	max_eval=100	-0.917	1052097
	5	max_eval=64	0.0	650000
	7	max_eval=64	0.0	650000

Table 3. Best obtained result for Analytic Function 1 (equation (3.1)). The 'best' result is the result with the lowest found function value. If multiple samples found the same value, the result with the fewest number of needed function evaluations is shown.

JHEP05(2021)108

Algorithm	dim	Parameters	min	N_{eval}
AMPGO	2		15.097	10062
	3		45.417	20000
	5		107.909	10032
	7		109.605	10456
CMA-ES	2	convergence=1.0000000000000001e-11, resolution=20.0	0.0	1500
	3	convergence=1.0000000000000001e-11, resolution=20.0	0.0	1780
	5	convergence=1.0000000000000001e-11, resolution=100.0	0.0	6900
	7	convergence=1.0000000000000001e-11, resolution=500.0	0.0	34500
Diver	2	convthresh=0.0001, np=10000	0.0	380000
	3	convthresh=0.0001, np=20000	0.0	1040000
	5	convthresh=0.0001, np=20000	0.0	1600000
	7	convthresh=0.0001, np=20000	0.0	2000000
Gaussian Particle Filter	2	logaritmic=False, survival_rate=0.5, width_decay=0.9	0.0	225589
	3	logaritmic=False, survival_rate=0.5, width_decay=0.95	0.0	965349
	5	logaritmic=False, survival_rate=0.2, width_decay=0.9	0.0	983262
	7	logaritmic=True, survival_rate=0.5, width_decay=0.95	3.266	977483
GPyOpt	2	eps=0.0001	0.002	754
	3	eps=0.0001	1.265	693
	5	eps=1e-06	5.284	587
	7	eps=0.001	5.829	797
Particle Swarm Optimisation	2	convthresh=0.001, np=5000	0.0	21200
	3	convthresh=0.001, np=2000	0.0	33600
	5	convthresh=0.0001, np=5000	0.0	85200
	7	convthresh=0.0001, np=2000	0.0	133600
PyGMO Artificial Bee Colony	2	generations=100, limit=50	0.0	4020
	3	generations=250, limit=100	0.0	10020
	5	generations=500, limit=100	0.0	20020
	7	generations=500, limit=50	0.0	20020
PyGMO Differential Evolution	2	generations=500, variant=iDE	0.0	1600
	3	generations=500, variant=iDE	0.0	1860
	5	generations=750, variant=iDE	0.0	4320
	7	generations=500, variant=jDE	0.0	6080
PyGMO Grey Wolf Optimisation	2	generations=100	0.0	2020
	3	generations=1000	0.0	20020
	5	generations=1000	0.0	20020
	7	generations=1000	0.0	20020
random sampling	2	n_samples=1000000	0.008	1000000
	3	n_samples=500000	0.512	500000
	5	n_samples=500000	5.873	500000
	7	n_samples=1000000	20.512	1000000
TuRBO	2	max_eval=100	0.0	1000584
	3	max_eval=100	0.027	1050660
	5	max_eval=100	2.044	1050025
	7	max_eval=100	12.101	1050007

Table 4. Best obtained result for Analytic Function 2 (equation (3.2)). The 'best' result is the result with the lowest found function value. If multiple samples found the same value, the result with the fewest number of needed function evaluations is shown.

JHEP05(2021)108

Algorithm	dim	Parameters	min	N_{eval}
AMPGO	2		-0.988	10014
	3		-0.699	20180
	5		-0.647	10008
	7		-0.447	20008
CMA-ES	2	convergence=1e-07, resolution=20.0	-1.0	760
	3	convergence=1e-07, resolution=20.0	-1.0	920
	5	convergence=0.0001, resolution=20.0	-1.0	1420
	7	convergence=1.0000000000000001e-11, resolution=20.0	-1.0	2340
Diver	2	convthresh=0.0001, np=10000	-1.0	210000
	3	convthresh=0.1, np=20000	-1.0	220000
	5	convthresh=0.0001, np=20000	-0.998	420000
	7	convthresh=0.1, np=10000	-0.984	110000
Gaussian Particle Filter	2	logaritmic=False, survival_rate=0.2, width_decay=0.9	-1.0	225589
	3	logaritmic=False, survival_rate=0.2, width_decay=0.9	-1.0	469460
	5	logaritmic=True, survival_rate=0.2, width_decay=0.9	-1.0	983184
	7	logaritmic=True, survival_rate=0.2, width_decay=0.9	-0.986	975284
GPyOpt	2	eps=1e-06	-1.0	636
	3	eps=0.01	-1.0	623
	5	eps=1e-06	-1.0	635
	7	eps=1e-05	-1.0	675
Particle Swarm Optimisation	2	convthresh=0.01, np=2000	-1.0	4400
	3	convthresh=0.0001, np=10000	-1.0	4400
	5	convthresh=0.0001, np=2000	-0.983	4400
	7	convthresh=0.001, np=5000	-0.923	4400
PyGMO Artificial Bee Colony	2	generations=100, limit=100	-1.0	4020
	3	generations=100, limit=50	-1.0	4020
	5	generations=500, limit=100	-1.0	20020
	7	generations=500, limit=100	-1.0	20020
PyGMO Differential Evolution	2	generations=750, variant=iDE	-1.0	2420
	3	generations=750, variant=iDE	-1.0	3140
	5	generations=250, variant=iDE	-1.0	5020
	7	generations=750, variant=jDE	-1.0	15020
PyGMO Grey Wolf Optimisation	2	generations=1000	-1.0	20020
	3	generations=100	-1.0	2020
	5	generations=1000	-1.0	20020
	7	generations=1000	-1.0	20020
random sampling	2	n_samples=50000	-1.0	50000
	3	n_samples=1000000	-1.0	1000000
	5	n_samples=1000000	-0.991	1000000
	7	n_samples=1000000	-0.964	1000000
TuRBO	2	max_eval=64	-1.0	700000
	3	max_eval=100	-1.0	1050981
	5	max_eval=100	-1.0	1050025
	7	max_eval=100	-0.998	1050000

Table 5. Best obtained result for Analytic Function 3 (equation (3.3)). The 'best' result is the result with the lowest found function value. If multiple samples found the same value, the result with the fewest number of needed function evaluations is shown.

JHEP05(2021)108

Algorithm	dim	Parameters	min	N_{eval}
AMPGO	2		302.982	20004
	3		576.415	20036
	5		1063.57	10086
	7		1874.14	20000
CMA-ES	2	convergence=0.0001, resolution=20.0	-0.0	1960
	3	convergence=1e-07, resolution=20.0	-0.0	3160
	5	convergence=1.0000000000000001e-11, resolution=20.0	-0.0	6800
	7	convergence=1.0000000000000001e-11, resolution=20.0	-0.0	4420
Diver	2	convthresh=0.0001, np=2000	-0.0	64000
	3	convthresh=0.0001, np=2000	-0.0	88000
	5	convthresh=0.0001, np=5000	-0.0	315000
	7	convthresh=0.0001, np=5000	-0.0	365000
Gaussian Particle Filter	2	logaritmic=True, survival_rate=0.2, width_decay=0.95	-0.0	463869
	3	logaritmic=True, survival_rate=0.2, width_decay=0.95	-0.0	965349
	5	logaritmic=False, survival_rate=0.5, width_decay=0.99	118.806	983262
	7	logaritmic=True, survival_rate=0.2, width_decay=0.9	493.412	977483
GPyOpt	2	eps=0.0001	0.017	651
	3	eps=0.01	0.062	852
	5	eps=1e-05	53.532	954
	7	eps=0.0001	928.87	1011
Particle Swarm Optimisation	2	convthresh=0.001, np=2000	-0.0	81600
	3	convthresh=0.001, np=20000	-0.0	110000
	5	convthresh=0.001, np=2000	-0.0	200400
	7	convthresh=0.0001, np=5000	-0.0	330000
PyGMO Artificial Bee Colony	2	generations=100, limit=100	-0.0	4020
	3	generations=250, limit=100	-0.0	10020
	5	generations=500, limit=100	-0.0	20020
	7	generations=750, limit=100	-0.0	30020
PyGMO Differential Evolution	2	generations=100, variant=jDE	-0.0	2020
	3	generations=250, variant=jDE	-0.0	5020
	5	generations=250, variant=iDE	-0.0	5020
	7	generations=500, variant=jDE	-0.0	10020
PyGMO Grey Wolf Optimisation	2	generations=1000	118.439	20020
	3	generations=1000	236.878	20020
	5	generations=1000	355.32	20020
	7	generations=50	1034.88	1020
random sampling	2	n_samples=500000	0.008	500000
	3	n_samples=500000	2.682	500000
	5	n_samples=1000000	184.329	1000000
	7	n_samples=500000	439.961	500000
TuRBO	2	max_eval=100	0.0	1000636
	3	max_eval=100	0.034	1050318
	5	max_eval=100	0.395	1050010
	7	max_eval=100	178.766	1050000

Table 6. Best obtained result for Analytic Function 4 (equation (3.4)). The 'best' result is the result with the lowest found function value. If multiple samples found the same value, the result with the fewest number of needed function evaluations is shown.

JHEP05(2021)108

Algorithm	Parameters	min	N_{eval}
AMPGO		262.815	10010
CMA-ES	convergence=500.0, resolution=0.0001	242.121	41000
Diver	convthresh=0.0001, np=20000	238.214	200000
Gaussian Particle Filter	logaritmic=True, survival_rate=0.5, width_decay=0.9	244.993	1333119
GPyOpt	eps=0.0001	255.827	684
Particle Swarm Optimisation	convthresh=0.001, np=20000	248.399	262800
PyGMO Artificial Bee Colony	generations=250, limit=50	242.189	10020
PyGMO Differential Evolution	generations=750, variant=2	242.197	15020
PyGMO Grey Wolf Optimisation	generations=1000	242.731	20020
random sampling	n_samples=1000000.0	245.489	1000000
TuRBO	max_evals=100	246.688	1000000

Table 7. Best obtained result for the approximation of the 12-dimensional MSSM7 log-likelihood described in section 3.2. The 'best' result is the result with the lowest found function value. If multiple samples found the same value, the result with the fewest number of needed function evaluations is shown.

Acknowledgments

MvB acknowledges support from the Science and Technology Facilities Council (grant number ST/T000864/1). R. RdA acknowledges partial funding/support from the Elusives ITN (Marie Skłodowska-Curie grant agreement No 674896), the "SOM Sabor y origen de la Materia" (FPA 2017-85985-P). BS, LH and SC acknowledges the support by the Netherlands eScience Center under the project iDark:The intelligent Dark Matter Survey.[9] AF is supported by an NSFC Research Fund for International Young Scientists grant 11950410509. MJW and CB are funded by the Australian Research Council (ARC) Discovery Project DP180102209, MJW and AL by the ARC Centre of Excellence for Dark Matter Particle Physics CE200100008, and PS by ARC Future Fellowship FT190100814. JM acknowledges the support from the grant PGC2018-094856-B-I00 by the Spanish MICIU / AEI and the European Union / FEDER, and the APOSTD/2019/165 fellowship by the University of Valencia, Generalitat Valenciana and European Union. EGM gratefully acknowledges the use of the facilities of Centro de Computación Científica (CCC) at Universidad Autónoma de Madrid. EGM also acknowledge financial support from Spanish Plan Nacional I+D+i, grants TIN2016-76406-P and PID2019-106827GB-I00 / AEI / 10.13039/501100011033.

References

[1] S.S. AbdusSalam et al., *Simple and statistically sound strategies for analysing physical theories*, arXiv:2012.09874 [INSPIRE].

[2] R.D. Cousins, *What is the likelihood function, and how is it used in particle physics?*, arXiv:2010.00356 [INSPIRE].

[9] https://www.esciencecenter.nl/projects/idark/.

JHEP05(2021)108

[3] G. Cowan, K. Cranmer, E. Gross and O. Vitells, *Asymptotic formulae for likelihood-based tests of new physics*, *Eur. Phys. J. C* **71** (2011) 1554 [*Erratum ibid.* **73** (2013) 2501] [arXiv:1007.1727] [INSPIRE].

[4] GAMBIT collaboration, *ColliderBit: a GAMBIT module for the calculation of high-energy collider observables and likelihoods*, *Eur. Phys. J. C* **77** (2017) 795 [arXiv:1705.07919] [INSPIRE].

[5] L.M. Rios and N.V. Sahinidis, *Derivative-free optimization: A review of algorithms and comparison of software implementations*, *J. Glob. Optim.* **56** (2012) 1247.

[6] F. James and M. Roos, *Minuit: A System for Function Minimization and Analysis of the Parameter Errors and Correlations*, *Comput. Phys. Commun.* **10** (1975) 343 [INSPIRE].

[7] F. Feroz, K. Cranmer, M. Hobson, R. Ruiz de Austri and R. Trotta, *Challenges of Profile Likelihood Evaluation in Multi-Dimensional SUSY Scans*, *JHEP* **06** (2011) 042 [arXiv:1101.3296] [INSPIRE].

[8] E.A. Baltz and P. Gondolo, *Markov chain Monte Carlo exploration of minimal supergravity with implications for dark matter*, *JHEP* **10** (2004) 052 [hep-ph/0407039] [INSPIRE].

[9] B.C. Allanach and C.G. Lester, *Multi-dimensional mSUGRA likelihood maps*, *Phys. Rev. D* **73** (2006) 015013 [hep-ph/0507283] [INSPIRE].

[10] R. Lafaye, T. Plehn and D. Zerwas, *SFITTER: SUSY parameter analysis at LHC and LC*, hep-ph/0404282 [INSPIRE].

[11] R. Ruiz de Austri, R. Trotta and L. Roszkowski, *A Markov chain Monte Carlo analysis of the CMSSM*, *JHEP* **05** (2006) 002 [hep-ph/0602028] [INSPIRE].

[12] C. Strege et al., *Profile likelihood maps of a 15-dimensional MSSM*, *JHEP* **09** (2014) 081 [arXiv:1405.0622] [INSPIRE].

[13] P. Bechtle et al., *Killing the CMSSM softly*, *Eur. Phys. J. C* **76** (2016) 96 [arXiv:1508.05951] [INSPIRE].

[14] M.E. Cabrera-Catalan, S. Ando, C. Weniger and F. Zandanel, *Indirect and direct detection prospect for TeV dark matter in the nine parameter MSSM*, *Phys. Rev. D* **92** (2015) 035018 [arXiv:1503.00599] [INSPIRE].

[15] K.J. de Vries et al., *The pMSSM10 after LHC Run 1*, *Eur. Phys. J. C* **75** (2015) 422 [arXiv:1504.03260] [INSPIRE].

[16] R. Trotta, R. Ruiz de Austri and L. Roszkowski, *Prospects for direct dark matter detection in the Constrained MSSM*, *New Astron. Rev.* **51** (2007) 316 [astro-ph/0609126] [INSPIRE].

[17] L. Roszkowski, R. Ruiz de Austri and R. Trotta, *Implications for the Constrained MSSM from a new prediction for $b \to s\gamma$*, *JHEP* **07** (2007) 075 [arXiv:0705.2012] [INSPIRE].

[18] L. Roszkowski, R. Ruiz de Austri, J. Silk and R. Trotta, *On prospects for dark matter indirect detection in the Constrained MSSM*, *Phys. Lett. B* **671** (2009) 10 [arXiv:0707.0622] [INSPIRE].

[19] G.D. Martinez, J.S. Bullock, M. Kaplinghat, L.E. Strigari and R. Trotta, *Indirect Dark Matter Detection from Dwarf Satellites: Joint Expectations from Astrophysics and Supersymmetry*, *JCAP* **06** (2009) 014 [arXiv:0902.4715] [INSPIRE].

[20] L. Roszkowski, R. Ruiz de Austri, R. Trotta, Y.-L.S. Tsai and T.A. Varley, *Global fits of the Non-Universal Higgs Model*, *Phys. Rev. D* **83** (2011) 015014 [*Erratum ibid.* **83** (2011) 039901] [arXiv:0903.1279] [INSPIRE].

JHEP05(2021)108

[21] L. Roszkowski, R. Ruiz de Austri and R. Trotta, *Efficient reconstruction of CMSSM parameters from LHC data: A case study*, *Phys. Rev. D* **82** (2010) 055003 [arXiv:0907.0594] [INSPIRE].

[22] P. Scott, J. Conrad, J. Edsjö, L. Bergström, C. Farnier and Y. Akrami, *Direct Constraints on Minimal Supersymmetry from Fermi-LAT Observations of the Dwarf Galaxy Segue 1*, *JCAP* **01** (2010) 031 [arXiv:0909.3300] [INSPIRE].

[23] Y. Akrami, P. Scott, J. Edsjö, J. Conrad and L. Bergström, *A Profile Likelihood Analysis of the Constrained MSSM with Genetic Algorithms*, *JHEP* **04** (2010) 057 [arXiv:0910.3950] [INSPIRE].

[24] G. Bertone, D.G. Cerdeño, M. Fornasa, R. Ruiz de Austri and R. Trotta, *Identification of Dark Matter particles with LHC and direct detection data*, *Phys. Rev. D* **82** (2010) 055008 [arXiv:1005.4280] [INSPIRE].

[25] Y. Akrami, C. Savage, P. Scott, J. Conrad and J. Edsjö, *How well will ton-scale dark matter direct detection experiments constrain minimal supersymmetry?*, *JCAP* **04** (2011) 012 [arXiv:1011.4318] [INSPIRE].

[26] M. Bridges, K. Cranmer, F. Feroz, M. Hobson, R. Ruiz de Austri and R. Trotta, *A Coverage Study of the CMSSM Based on ATLAS Sensitivity Using Fast Neural Networks Techniques*, *JHEP* **03** (2011) 012 [arXiv:1011.4306] [INSPIRE].

[27] G. Bertone, D. Cumberbatch, R. Ruiz de Austri and R. Trotta, *Dark Matter Searches: The Nightmare Scenario*, *JCAP* **01** (2012) 004 [arXiv:1107.5813] [INSPIRE].

[28] G. Bertone, D.G. Cerdeño, M. Fornasa, L. Pieri, R. Ruiz de Austri and R. Trotta, *Complementarity of Indirect and Accelerator Dark Matter Searches*, *Phys. Rev. D* **85** (2012) 055014 [arXiv:1111.2607] [INSPIRE].

[29] Y. Akrami, C. Savage, P. Scott, J. Conrad and J. Edsjö, *Statistical coverage for supersymmetric parameter estimation: a case study with direct detection of dark matter*, *JCAP* **07** (2011) 002 [arXiv:1011.4297] [INSPIRE].

[30] IceCube collaboration, *Use of event-level neutrino telescope data in global fits for theories of new physics*, *JCAP* **11** (2012) 057 [arXiv:1207.0810] [INSPIRE].

[31] G. Bertone, D.G. Cerdeño, M. Fornasa, R. Ruiz de Austri, C. Strege and R. Trotta, *Global fits of the CMSSM including the first LHC and XENON100 data*, *JCAP* **01** (2012) 015 [arXiv:1107.1715] [INSPIRE].

[32] G. Bertone et al., *Global analysis of the pMSSM in light of the Fermi GeV excess: prospects for the LHC Run-II and astroparticle experiments*, *JCAP* **04** (2016) 037 [arXiv:1507.07008] [INSPIRE].

[33] O. Buchmueller et al., *Predictions for Supersymmetric Particle Masses using Indirect Experimental and Cosmological Constraints*, *JHEP* **09** (2008) 117 [arXiv:0808.4128] [INSPIRE].

[34] O. Buchmueller et al., *Likelihood Functions for Supersymmetric Observables in Frequentist Analyses of the CMSSM and NUHM1*, *Eur. Phys. J. C* **64** (2009) 391 [arXiv:0907.5568] [INSPIRE].

[35] O. Buchmueller et al., *Frequentist Analysis of the Parameter Space of Minimal Supergravity*, *Eur. Phys. J. C* **71** (2011) 1583 [arXiv:1011.6118] [INSPIRE].

JHEP05(2021)108

[36] O. Buchmueller et al., *Implications of Initial LHC Searches for Supersymmetry*, *Eur. Phys. J. C* **71** (2011) 1634 [arXiv:1102.4585] [INSPIRE].

[37] O. Buchmueller et al., *Supersymmetry and Dark Matter in Light of LHC 2010 and Xenon100 Data*, *Eur. Phys. J. C* **71** (2011) 1722 [arXiv:1106.2529] [INSPIRE].

[38] O. Buchmueller et al., *Higgs and Supersymmetry*, *Eur. Phys. J. C* **72** (2012) 2020 [arXiv:1112.3564] [INSPIRE].

[39] O. Buchmueller et al., *The NUHM2 after LHC Run 1*, *Eur. Phys. J. C* **74** (2014) 3212 [arXiv:1408.4060] [INSPIRE].

[40] E. Bagnaschi et al., *Likelihood Analysis of Supersymmetric* SU(5) *GUTs*, *Eur. Phys. J. C* **77** (2017) 104 [arXiv:1610.10084] [INSPIRE].

[41] E. Bagnaschi et al., *Likelihood Analysis of the Minimal AMSB Model*, *Eur. Phys. J. C* **77** (2017) 268 [arXiv:1612.05210] [INSPIRE].

[42] B.C. Allanach, K. Cranmer, C.G. Lester and A.M. Weber, *Natural priors, CMSSM fits and LHC weather forecasts*, *JHEP* **08** (2007) 023 [arXiv:0705.0487] [INSPIRE].

[43] S.S. AbdusSalam, B.C. Allanach, F. Quevedo, F. Feroz and M. Hobson, *Fitting the Phenomenological MSSM*, *Phys. Rev. D* **81** (2010) 095012 [arXiv:0904.2548] [INSPIRE].

[44] S.S. AbdusSalam, B.C. Allanach, M.J. Dolan, F. Feroz and M.P. Hobson, *Selecting a Model of Supersymmetry Breaking Mediation*, *Phys. Rev. D* **80** (2009) 035017 [arXiv:0906.0957] [INSPIRE].

[45] B.C. Allanach, *Impact of CMS Multi-jets and Missing Energy Search on CMSSM Fits*, *Phys. Rev. D* **83** (2011) 095019 [arXiv:1102.3149] [INSPIRE].

[46] B.C. Allanach, T.J. Khoo, C.G. Lester and S.L. Williams, *The impact of the ATLAS zero-lepton, jets and missing momentum search on a CMSSM fit*, *JHEP* **06** (2011) 035 [arXiv:1103.0969] [INSPIRE].

[47] C. Balázs, A. Buckley, D. Carter, B. Farmer and M. White, *Should we still believe in constrained supersymmetry?*, *Eur. Phys. J. C* **73** (2013) 2563 [arXiv:1205.1568] [INSPIRE].

[48] M.E. Cabrera, J.A. Casas and R. Ruiz de Austri, *The health of SUSY after the Higgs discovery and the XENON100 data*, *JHEP* **07** (2013) 182 [arXiv:1212.4821] [INSPIRE].

[49] A. Fowlie, K. Kowalska, L. Roszkowski, E.M. Sessolo and Y.-L.S. Tsai, *Dark matter and collider signatures of the MSSM*, *Phys. Rev. D* **88** (2013) 055012 [arXiv:1306.1567] [INSPIRE].

[50] S. Henrot-Versillé et al., *Constraining Supersymmetry using the relic density and the Higgs boson*, *Phys. Rev. D* **89** (2014) 055017 [arXiv:1309.6958] [INSPIRE].

[51] D. Kim, P. Athron, C. Balázs, B. Farmer and E. Hutchison, *Bayesian naturalness of the CMSSM and CNMSSM*, *Phys. Rev. D* **90** (2014) 055008 [arXiv:1312.4150] [INSPIRE].

[52] K. Kowalska, L. Roszkowski, E.M. Sessolo and A.J. Williams, *GUT-inspired SUSY and the muon $g-2$ anomaly: prospects for LHC 14 TeV*, *JHEP* **06** (2015) 020 [arXiv:1503.08219] [INSPIRE].

[53] M.E. Cabrera, J.A. Casas, A. Delgado, S. Robles and R. Ruiz de Austri, *Naturalness of MSSM dark matter*, *JHEP* **08** (2016) 058 [arXiv:1604.02102] [INSPIRE].

[54] C. Han, K.-i. Hikasa, L. Wu, J.M. Yang and Y. Zhang, *Status of CMSSM in light of current LHC Run-2 and LUX data*, *Phys. Lett. B* **769** (2017) 470 [arXiv:1612.02296] [INSPIRE].

JHEP05(2021)108

[55] P. Bechtle et al., *How alive is constrained SUSY really?*, *Nucl. Part. Phys. Proc.* **273–275** (2016) 589 [arXiv:1410.6035] [INSPIRE].

[56] L. Roszkowski, E.M. Sessolo and A.J. Williams, *What next for the CMSSM and the NUHM: Improved prospects for superpartner and dark matter detection*, *JHEP* **08** (2014) 067 [arXiv:1405.4289] [INSPIRE].

[57] A. Fowlie and M. Raidal, *Prospects for constrained supersymmetry at* $\sqrt{s} = 33$ *TeV and* $\sqrt{s} = 100$ *TeV proton-proton super-colliders*, *Eur. Phys. J. C* **74** (2014) 2948 [arXiv:1402.5419] [INSPIRE].

[58] O. Buchmueller et al., *The CMSSM and NUHM1 after LHC Run 1*, *Eur. Phys. J. C* **74** (2014) 2922 [arXiv:1312.5250] [INSPIRE].

[59] O. Buchmueller et al., *Implications of Improved Higgs Mass Calculations for Supersymmetric Models*, *Eur. Phys. J. C* **74** (2014) 2809 [arXiv:1312.5233] [INSPIRE].

[60] P. Bechtle et al., *Constrained Supersymmetry after the Higgs Boson Discovery: A global analysis with Fittino*, *PoS* **EPS-HEP2013** (2013) 313 [arXiv:1310.3045] [INSPIRE].

[61] N. Bornhauser and M. Drees, *Determination of the CMSSM Parameters using Neural Networks*, *Phys. Rev. D* **88** (2013) 075016 [arXiv:1307.3383] [INSPIRE].

[62] S. Akula and P. Nath, *Gluino-driven radiative breaking, Higgs boson mass, muon g-2, and the Higgs diphoton decay in supergravity unification*, *Phys. Rev. D* **87** (2013) 115022 [arXiv:1304.5526] [INSPIRE].

[63] M. Citron, J. Ellis, F. Luo, J. Marrouche, K.A. Olive and K.J. de Vries, *End of the CMSSM coannihilation strip is nigh*, *Phys. Rev. D* **87** (2013) 036012 [arXiv:1212.2886] [INSPIRE].

[64] C. Strege, G. Bertone, F. Feroz, M. Fornasa, R. Ruiz de Austri and R. Trotta, *Global Fits of the CMSSM and NUHM including the LHC Higgs discovery and new XENON100 constraints*, *JCAP* **04** (2013) 013 [arXiv:1212.2636] [INSPIRE].

[65] A.V. Gladyshev and D.I. Kazakov, *Is (Low Energy) SUSY Still Alive?*, in *2012 European School of High-Energy Physics*, (2012), DOI [arXiv:1212.2548] [INSPIRE].

[66] K. Kowalska, S. Munir, L. Roszkowski, E.M. Sessolo, S. Trojanowski and Y.-L.S. Tsai, *Constrained next-to-minimal supersymmetric standard model with a 126 GeV Higgs boson: A global analysis*, *Phys. Rev. D* **87** (2013) 115010 [arXiv:1211.1693] [INSPIRE].

[67] O. Buchmueller et al., *The CMSSM and NUHM1 in Light of 7 TeV LHC,* $B_s \to \mu^+\mu^-$ *and XENON100 Data*, *Eur. Phys. J. C* **72** (2012) 2243 [arXiv:1207.7315] [INSPIRE].

[68] S. Akula, P. Nath and G. Peim, *Implications of the Higgs Boson Discovery for mSUGRA*, *Phys. Lett. B* **717** (2012) 188 [arXiv:1207.1839] [INSPIRE].

[69] H. Baer, V. Barger, A. Lessa and X. Tata, *Discovery potential for SUSY at a high luminosity upgrade of LHC14*, *Phys. Rev. D* **86** (2012) 117701 [arXiv:1207.4846] [INSPIRE].

[70] L. Roszkowski, E.M. Sessolo and Y.-L.S. Tsai, *Bayesian Implications of Current LHC Supersymmetry and Dark Matter Detection Searches for the Constrained MSSM*, *Phys. Rev. D* **86** (2012) 095005 [arXiv:1202.1503] [INSPIRE].

[71] C. Strege, G. Bertone, D.G. Cerdeño, M. Fornasa, R. Ruiz de Austri and R. Trotta, *Updated global fits of the CMSSM including the latest LHC SUSY and Higgs searches and XENON100 data*, *JCAP* **03** (2012) 030 [arXiv:1112.4192] [INSPIRE].

JHEP05(2021)108

[72] P. Bechtle et al., *Constrained Supersymmetry after two years of LHC data: a global view with Fittino*, *JHEP* **06** (2012) 098 [arXiv:1204.4199] [INSPIRE].

[73] O. Buchmueller et al., *Supersymmetry in Light of 1/fb of LHC Data*, *Eur. Phys. J. C* **72** (2012) 1878 [arXiv:1110.3568] [INSPIRE].

[74] A. Fowlie, A. Kalinowski, M. Kazana, L. Roszkowski and Y.L.S. Tsai, *Bayesian Implications of Current LHC and XENON100 Search Limits for the Constrained MSSM*, *Phys. Rev. D* **85** (2012) 075012 [arXiv:1111.6098] [INSPIRE].

[75] P. Bechtle, K. Desch, M. Uhlenbrock and P. Wienemann, *Constraining SUSY models with Fittino using measurements before, with and beyond the LHC*, *Eur. Phys. J. C* **66** (2010) 215 [arXiv:0907.2589] [INSPIRE].

[76] R. Trotta, F. Feroz, M.P. Hobson, L. Roszkowski and R. Ruiz de Austri, *The impact of priors and observables on parameter inferences in the Constrained MSSM*, *JHEP* **12** (2008) 024 [arXiv:0809.3792] [INSPIRE].

[77] P. Bechtle, K. Desch and P. Wienemann, *Fittino, a program for determining MSSM parameters from collider observables using an iterative method*, *Comput. Phys. Commun.* **174** (2006) 47 [hep-ph/0412012] [INSPIRE].

[78] K. Kowalska, *Phenomenological MSSM in light of new 13 TeV LHC data*, *Eur. Phys. J. C* **76** (2016) 684 [arXiv:1608.02489] [INSPIRE].

[79] E.A. Bagnaschi et al., *Supersymmetric Dark Matter after LHC Run 1*, *Eur. Phys. J. C* **75** (2015) 500 [arXiv:1508.01173] [INSPIRE].

[80] S.S. AbdusSalam and L. Velasco-Sevilla, *Where to look for natural supersymmetry*, *Phys. Rev. D* **94** (2016) 035026 [arXiv:1506.02499] [INSPIRE].

[81] E. Bagnaschi et al., *Likelihood Analysis of the pMSSM11 in Light of LHC 13-TeV Data*, *Eur. Phys. J. C* **78** (2018) 256 [arXiv:1710.11091] [INSPIRE].

[82] K. Cheung, Y.-L.S. Tsai, P.-Y. Tseng, T.-C. Yuan and A. Zee, *Global Study of the Simplest Scalar Phantom Dark Matter Model*, *JCAP* **10** (2012) 042 [arXiv:1207.4930] [INSPIRE].

[83] A. Arhrib, Y.-L.S. Tsai, Q. Yuan and T.-C. Yuan, *An Updated Analysis of Inert Higgs Doublet Model in light of the Recent Results from LUX, PLANCK, AMS-02 and LHC*, *JCAP* **06** (2014) 030 [arXiv:1310.0358] [INSPIRE].

[84] S. Matsumoto, S. Mukhopadhyay and Y.-L.S. Tsai, *Singlet Majorana fermion dark matter: a comprehensive analysis in effective field theory*, *JHEP* **10** (2014) 155 [arXiv:1407.1859] [INSPIRE].

[85] D. Chowdhury and O. Eberhardt, *Global fits of the two-loop renormalized Two-Higgs-Doublet model with soft Z_2 breaking*, *JHEP* **11** (2015) 052 [arXiv:1503.08216] [INSPIRE].

[86] S. Liem et al., *Effective field theory of dark matter: a global analysis*, *JHEP* **09** (2016) 077 [arXiv:1603.05994] [INSPIRE].

[87] X. Huang, Y.-L.S. Tsai and Q. Yuan, *LikeDM: likelihood calculator of dark matter detection*, *Comput. Phys. Commun.* **213** (2017) 252 [arXiv:1603.07119] [INSPIRE].

[88] S. Banerjee, S. Matsumoto, K. Mukaida and Y.-L.S. Tsai, *WIMP Dark Matter in a Well-Tempered Regime: A case study on Singlet-Doublets Fermionic WIMP*, *JHEP* **11** (2016) 070 [arXiv:1603.07387] [INSPIRE].

JHEP05(2021)108

[89] S. Matsumoto, S. Mukhopadhyay and Y.-L.S. Tsai, *Effective Theory of WIMP Dark Matter supplemented by Simplified Models: Singlet-like Majorana fermion case*, *Phys. Rev. D* **94** (2016) 065034 [arXiv:1604.02230] [INSPIRE].

[90] A. Cuoco, B. Eiteneuer, J. Heisig and M. Krämer, *A global fit of the γ-ray galactic center excess within the scalar singlet Higgs portal model*, *JCAP* **06** (2016) 050 [arXiv:1603.08228] [INSPIRE].

[91] V. Cacchio, D. Chowdhury, O. Eberhardt and C.W. Murphy, *Next-to-leading order unitarity fits in Two-Higgs-Doublet models with soft $\mathbb{Z}_2$ breaking*, *JHEP* **11** (2016) 026 [arXiv:1609.01290] [INSPIRE].

[92] G. Bertone, K. Kong, R. Ruiz de Austri and R. Trotta, *Global fits of the Minimal Universal Extra Dimensions scenario*, *Phys. Rev. D* **83** (2011) 036008 [arXiv:1010.2023] [INSPIRE].

[93] C.-W. Chiang, G. Cottin and O. Eberhardt, *Global fits in the Georgi-Machacek model*, *Phys. Rev. D* **99** (2019) 015001 [arXiv:1807.10660] [INSPIRE].

[94] J. De Blas et al., *HEPfit: a code for the combination of indirect and direct constraints on high energy physics models*, *Eur. Phys. J. C* **80** (2020) 456 [arXiv:1910.14012] [INSPIRE].

[95] S. Matsumoto, Y.-L.S. Tsai and P.-Y. Tseng, *Light Fermionic WIMP Dark Matter with Light Scalar Mediator*, *JHEP* **07** (2019) 050 [arXiv:1811.03292] [INSPIRE].

[96] F. Feroz, M.P. Hobson, E. Cameron and A.N. Pettitt, *Importance Nested Sampling and the MultiNest Algorithm*, *Open J. Astrophys.* **2** (2019) 10 [arXiv:1306.2144] [INSPIRE].

[97] F. Feroz, M.P. Hobson and M. Bridges, *MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics*, *Mon. Not. Roy. Astron. Soc.* **398** (2009) 1601 [arXiv:0809.3437] [INSPIRE].

[98] F. Feroz and M.P. Hobson, *Multimodal nested sampling: an efficient and robust alternative to MCMC methods for astronomical data analysis*, *Mon. Not. Roy. Astron. Soc.* **384** (2008) 449 [arXiv:0704.3704] [INSPIRE].

[99] J. Dunkley, M. Bucher, P.G. Ferreira, K. Moodley and C. Skordis, *Fast and reliable mcmc for cosmological parameter estimation*, *Mon. Not. Roy. Astron. Soc.* **356** (2005) 925 [astro-ph/0405462] [INSPIRE].

[100] A. Putze and L. Derome, *The Grenoble Analysis Toolkit (GreAT) — A statistical analysis framework*, *Phys. Dark Univ.* **5** (2014) 29.

[101] W.K. Hastings, *Monte carlo sampling methods using markov chains and their applications*, *Biometrika* **57** (1970) 97.

[102] M. van Beekveld, S. Caron and R. Ruiz de Austri, *The current status of fine-tuning in supersymmetry*, *JHEP* **01** (2020) 147 [arXiv:1906.10706] [INSPIRE].

[103] GAMBIT collaboration, *A global fit of the MSSM with GAMBIT*, *Eur. Phys. J. C* **77** (2017) 879 [arXiv:1705.07917] [INSPIRE].

[104] D.W. Hogg and D. Foreman-Mackey, *Data analysis recipes: Using Markov Chain Monte Carlo*, *Astrophys. J. Suppl.* **236** (2018) 11 [arXiv:1710.06068] [INSPIRE].

[105] J. Skilling, *Nested sampling for general Bayesian computation*, *Bayesian Anal.* **1** (2006) 833.

[106] A. Kvellestad, P. Scott and M. White, *GAMBIT and its Application in the Search for Physics Beyond the Standard Model*, arXiv:1912.04079 [INSPIRE].

JHEP05(2021)108

[107] S. Hoof, F. Kahlhoefer, P. Scott, C. Weniger and M. White, *Axion global fits with Peccei-Quinn symmetry breaking before inflation using GAMBIT*, *JHEP* **03** (2019) 191 [*Erratum ibid.* **11** (2019) 099] [arXiv:1810.07192] [INSPIRE].

[108] GAMBIT collaboration, *Global analyses of Higgs portal singlet dark matter models using GAMBIT*, *Eur. Phys. J. C* **79** (2019) 38 [arXiv:1808.10465] [INSPIRE].

[109] GAMBIT collaboration, *Global fits of GUT-scale SUSY models with GAMBIT*, *Eur. Phys. J. C* **77** (2017) 824 [arXiv:1705.07935] [INSPIRE].

[110] GAMBIT collaboration, *Combined collider constraints on neutralinos and charginos*, *Eur. Phys. J. C* **79** (2019) 395 [arXiv:1809.02097] [INSPIRE].

[111] GAMBIT collaboration, *Comparison of statistical sampling methods with ScannerBit, the GAMBIT scanning module*, *Eur. Phys. J. C* **77** (2017) 761 [arXiv:1705.07959] [INSPIRE].

[112] R. Storn and K. Price, *Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces*, *J. Glob. Optim.* **11** (1997) 341.

[113] K. Price, R.M. Storn, and J.A. Lampinen, *Differential evolution: a practical approach to global optimization*, Springer, (2005).

[114] S. Das and P. Suganthan, *Differential evolution: A survey of the state-of-the-art*, *IEEE Trans. Evol. Comput.* **15** (2011) 4.

[115] K. Price, *Differential evolution*, in *Handbook of Optimization*, I. Zelinka, V. Snášel, and A. Abraham, eds., vol. 38 of *Intelligent Systems Reference Library*, pp. 187–214. Springer Berlin Heidelberg, (2013).

[116] J. Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, *Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems*, *IEEE Trans. Evol. Comput.* **10** (2006) 646.

[117] GAMBIT collaboration, *GAMBIT: The Global and Modular Beyond-the-Standard-Model Inference Tool*, *Eur. Phys. J. C* **77** (2017) 784 [*Addendum ibid.* **78** (2018) 98] [arXiv:1705.07908] [INSPIRE].

[118] S.M. Elsayed, R.A. Sarker and D.L. Essam, *United multi-operator evolutionary algorithms*, in *2014 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, (Jul. 2014), DOI.

[119] D.I.F. Biscani, *Pygmo (1.1.7)*, (2015).

[120] J. Kennedy and R. Eberhart, *Particle swarm optimization*, in *Proceedings of ICNN'95 — International Conference on Neural Networks*, vol. 4, (1995), pp. 1942–1948.

[121] M.R. Bonyadi and Z. Michalewicz, *Particle swarm optimization for single objective continuous space problems: A review*, *Evol. Comput.* **25** (2017) 1.

[122] N. Hansen, *The CMA Evolution Strategy: A Comparing Review*, Springer Berlin Heidelberg, Berlin, Heidelberg, (2006), pp. 75–102.

[123] J. Snoek, H. Larochelle and R. Adams, *Practical Bayesian optimization of machine learning algorithms*, arXiv:1206.2944.

[124] A. Bull, *Convergence rates of efficient global optimization algorithms*, arXiv:1101.3501.

[125] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, Massachusetts Institute of Technology, (2006).

[126] D. Eriksson, M. Pearce, J.R. Gardner, R. Turner and M. Poloczek, *Scalable global optimization via local bayesian optimization*, arXiv:1910.01739.

JHEP05(2021)108

[127] S. Mirjalili, S. Mirjalili and A. Lewis, *Grey wolf optimizer*, *Adv. Eng. Softw.* **69** (2014) 46.

[128] D. Karaboga, *An idea based on honey bee swarm for numerical optimization*, TECHNICAL REPORT-TR06, Erciyes University, Turkey (2005).

[129] B. Akay and D. Karaboga, *A modified artificial bee colony algorithm for real-parameter optimization*, *Info. Sci.* **192** (2012) 120.

[130] M. Mernik, S.-H. Liu, D. Karaboga and M. Črepinšek, *On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation*, *Info. Sci.* **291** (2015) 115.

[131] J.H. Kotecha and P.M. Djuric, *Gaussian particle filtering*, *IEEE Trans. Signal Processing* **51** (2003) 2592.

[132] L. Lasdon, A. Duarte, F. Glover, M. Laguna and R. Marti, *Adaptive memory programming for constrained global optimization*, *Comput. Oper. Res.* **37** (2010) 1500.

[133] A. Levy and S. Gómez, *The tunneling method applied to global optimization*, in *Numerical Optimization*, SIAM (1985), pp. 213–244.

[134] R.H. Byrd, P. Lu, J. Nocedal and C. Zhu, *A limited-memory algorithm for bound constrained optimization*, *SIAM J. Sci. Comput.* **16** (1994) 1190.

[135] *Ampgo analysis*, http://infinity77.net/global_optimization/ampgo.html, Accessed: 01 April 2020.

[136] G. Brooijmans et al., *Les Houches 2019 Physics at TeV Colliders: New Physics Working Group Report*, in *11th Les Houches Workshop on Physics at TeV Colliders: PhysTeV Les Houches*, (2020) [arXiv:2002.12220] [INSPIRE].

[137] A. Buckley, A. Shilton and M.J. White, *Fast supersymmetry phenomenology at the Large Hadron Collider using machine learning techniques*, *Comput. Phys. Commun.* **183** (2012) 960 [arXiv:1106.4613] [INSPIRE].

[138] G. Klambauer, T. Unterthiner, A. Mayr and S. Hochreiter, *Self-normalizing neural networks*, arXiv:1706.02515.

[139] D.P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv:1412.6980.

JHEP05(2021)108